MORE ON - ArduinoBots, Big Walkers, CPLDs

SERVO MAGAZINE

FOR THE ROBOT INNOVATOR

www.servomagazine.com

April 2011

Stepper Motor Control EXPERIMENTS With VEX

Build the SunBot III

✦ DARwIn-OP

Fully open source hardware and software platform from Virginia Tech, University of Pennsylvania, Purdue University, and Robotis

✦ Combat Zone

SO YOU WANT TO FIGHT ROBOTS?

Turning Your Destructive Instincts into a Socially Acceptable Hobby!

U.S. $5.50 CANADA $7.00

04

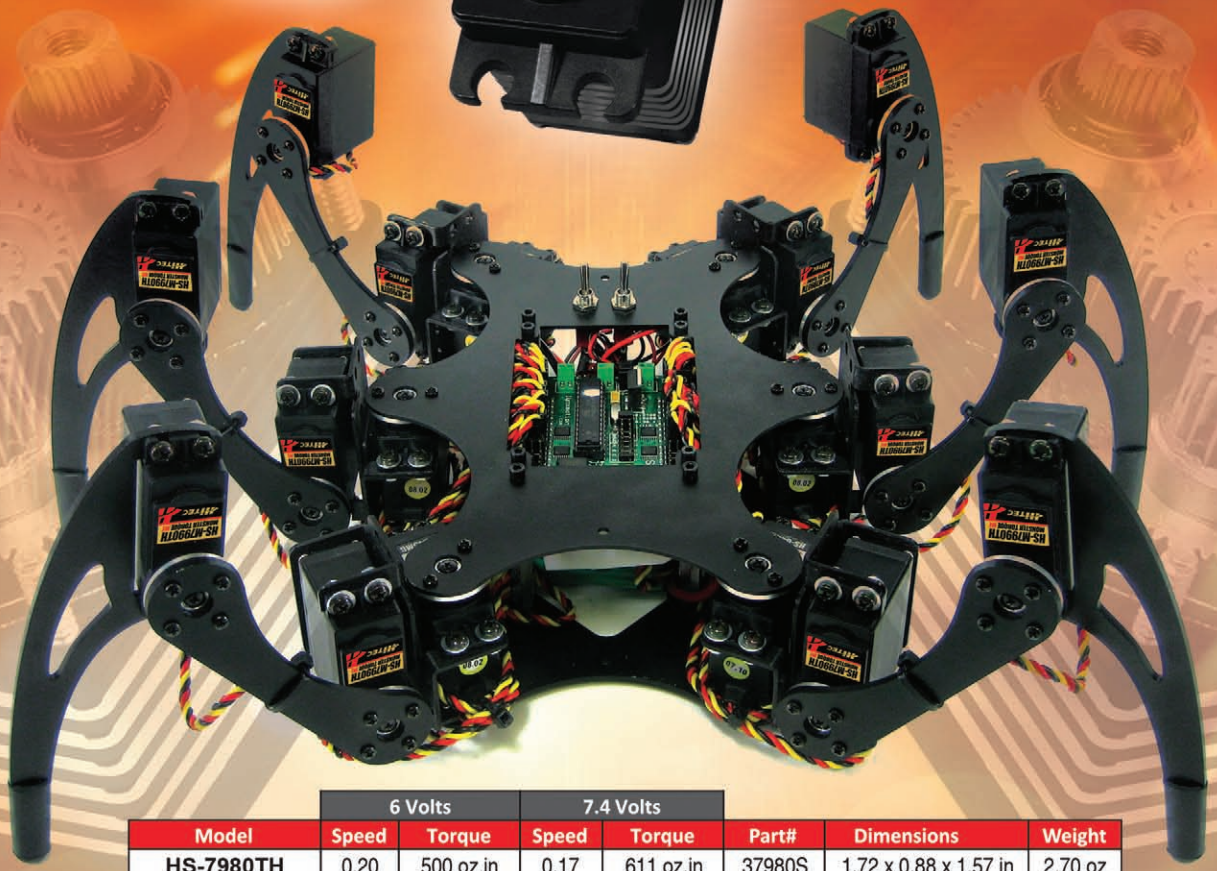
0 71486 02422 4

INVEST in the BEST

YOUR ROBOT WILL THANK YOU

At Hitec, we know how much time and money you dedicate to your robot hobby. So why not make sure your servos are delivering what they promise? Our technologically advanced mega servos are tough enough for even your most challenging projects. Designed with our newest high resolution "G2.5" 12-bit programmable digital circuit and indestructible titanium gears, the HS-7980TH and HS-M7990TH give mega torque and speed with pinpoint accuracy. Built to last, these servos bring unprecedented power and sustainability to your investment.

Our HS-7980TH employs durable mechanical potentiometer technology while the HS-M7990TH utilizes the first ever magnetic encoder.



	6 Volts		7.4 Volts				
Model	Speed	Torque	Speed	Torque	Part#	Dimensions	Weight
HS-7980TH	0.20	500 oz.in	0.17	611 oz.in	37980S	1.72 x 0.88 x 1.57 in	2.70 oz
HS-M7990TH	0.20	500 oz.in	0.17	611 oz.in	37990S	1.72 x 0.88 x 1.57 in	2.70 oz

HITEC

We Serve you Right!

12115 Paine Street, Poway, CA 92064 • 858-748-6948 • www.hitecrcd.com

Thanks to Lynxmotion for the Phoenix robot featured in this advertisement.

Take your design from idea to reality

Pololu
Engage Your Brain

www.pololu.com

SERVO MAGAZINE

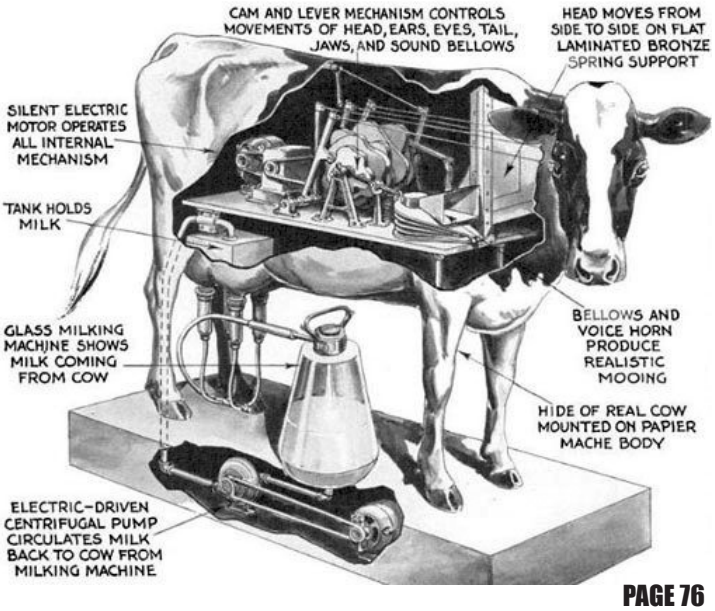
04.2011

VOL. 9 NO. 4

Did you know that each article in *SERVO Magazine* has its own webpage? It's where you go for downloads, comments, updates, corrections, or to link to the article in the digital issue. The unique link for each webpage is included with the article.

You can also visit article pages from back issues at **www.servomagazine.com**. Just select the ***Past Issues*** tab from the ***About SERVO*** drop down menu, click the Table of Contents link, and the article name.

Columns



PAGE 76

Features

Events

The Combat Zone...

PAGE 26

***SERVO** Magazine* **(ISSN 1546-0592/CDN Pub Agree#40702530)** is published monthly for $24.95 per year by T & L Publications, Inc., 430 Princeland Court, Corona, CA 92879. PERIODICALS POSTAGE PAID AT CORONA, CA AND AT ADDITIONAL ENTRY MAILING OFFICES. POSTMASTER: Send address changes to **SERVO Magazine, P.O. Box 15277, North Hollywood, CA 91615** or Station A, P.O. Box 54, Windsor ON N9A 6J5; cpcreturns@servomagazine.com

In This Issue ...

Departments

Mind / Iron

by Bryan Bergeron, Editor

3D Printers and Self-Replicating Machines

When the Apple LaserWriter first appeared on the market in 1985, the $5,000 laser printer and Macintosh computer helped to create the self-publishing industry. The LaserWriter and other desktop laser printers enabled small business owners and well-heeled computer enthusiasts to produce professional documents on their desktop.

Today, 3D printers are at about the same price point, with commercial models available for less than $10K. Although supplies are still expensive, this new price point is a significant improvement over the $30K - $50K required to buy entry-level technology only a few years ago. Moreover, there are a number of do-it-yourself 3D printers available, including the open source Cupcake CNC. Take a look at the Wikipedia entry for 3D printers for sample 3D objects and links to affordable DIY printers. From a robotics perspective, the 3D printers are a blend of servos, controllers, electronic valves, and software drivers that resemble ink-jet printing technology on steroids. Instead of spraying ink, these printers spray layer upon layer of plastic material. By laying down hundreds of layers or slices — as defined by 3D software — these printers can create small plastic objects of just about anything that can be imagined. Some experimenters have even used 3D printers to create parts for other 3D printers. Not quite self-replicating, but it's getting there.

While the robotics technology in the machines is worthy of study, what these 3D printers produce is also exciting. If you're into small robots, you can print just about any structural components you need. The technology isn't quite there if you need to create small gears with fine teeth, but for wheels, chassis, mounts, and other plastic components, if you can draw it in a 3D graphics program, you can print it.

Like the early laser printers, I expect that the first wave of serious robotics use will be with groups. As a shared resource, say purchased by a robotics club, even entry-level commercial 3D printers may be affordable. However, if your goal isn't to build miniature carpet crawlers but to explore the robotics involved in 3D printing, then an open source DIY printer is an option even for a personal project. Although you may be able to build your own 3D printer from scratch, a big issue is the 3D software that drives the printers. Unless you're a 3D graphics programming wiz, you'll want to be compatible with the drivers and other software that's been developed for DIY 3D printers.

For now, you'll need conventional motors, printed circuit boards, and other non-plastic parts to finish off your robots. This may change soon, given the work in conductive plastics and with printable circuits based on standard ink-jet technology. Take a look at 'printed electronics' on Wikipedia for a good introduction to the technology.

I expect that in the future, the digital version of *SERVO* will include source files for printing complete robots. Until then, if you're experimenting with 3D printing, please consider sharing your experiences with your fellow readers. SV

FOR THE ROBOT INNOVATOR SERVO MAGAZINE

Published Monthly By
T & L Publications, Inc.
430 Princeland Ct., Corona, CA 92879-1300
(951) 371-8497
FAX **(951) 371-3052**
Webstore Only **1-800-783-4624**
www.servomagazine.com

Subscriptions
Toll Free **1-877-525-2539**
Outside US **1-818-487-4545**
P.O. Box 15277, N. Hollywood, CA 91615

PUBLISHER
Larry Lemieux
publisher@servomagazine.com

ASSOCIATE PUBLISHER/ VP OF SALES/MARKETING
Robin Lemieux
display@servomagazine.com

EDITOR
Bryan Bergeron
techedit-servo@yahoo.com

CONTRIBUTING EDITORS
Jeff Eckert, Jenn Eckert
Tom Carroll, David Geer
Dennis Clark, R. Steven Rainwater
Fred Eady, Kevin Berry
Gregory Intermaggio, Gordon McComb
David Ward, Daniel Albert
Daniel Ramirez, Mike Jeffries
Collin Berry, Thomas Kenney

CIRCULATION DIRECTOR
Tracy Kerley
subscribe@servomagazine.com

MARKETING COORDINATOR WEBSTORE
Brian Kirkpatrick
sales@servomagazine.com

WEB CONTENT
Michael Kaudze
website@servomagazine.com

ADMINISTRATIVE ASSISTANT
Debbie Stauffacher

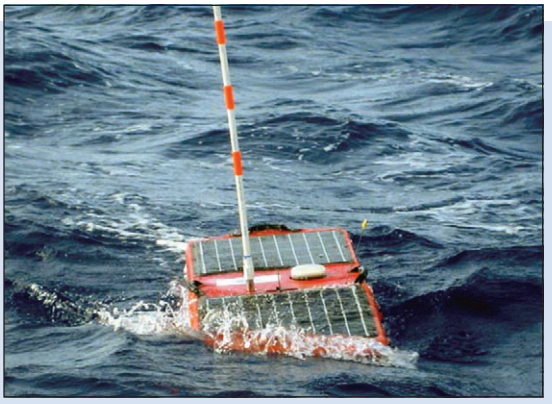
PRODUCTION/GRAPHICS
Shannon Christensen


Robytes

by Jeff and Jenn Eckert

UMV Taps Ambient Power

Late late last year, Liquid Robotics, Inc., received the *Wall Street Journal's* Innovation Award in the robotics category for its Wave Glider unmanned maritime vehicle (UMV), and it indeed is a slick design. Since their initial introduction in 2008, the gliders have logged more than 100,000 miles (185,000 km) of travel, all without any need for refueling. The design uses a floating platform coupled with a submerged glider with wing-shaped fins to convert wave motion into forward thrust. It also incorporates solar panels to provide electricity to its payload of sensors and communication equipment. This allows it to travel over vast areas while collecting data and communicating with shore facilities via satellite. Initially developed to monitor humpback whales, it is reconfigurable for such applications as port and harbor security, maritime law enforcement, meteorology, pollution detection, and so forth. In fact, BP deployed two of them to monitor water quality near the Deepwater Horizon disaster site. For details on some of the UMV's voyages — including a circumnavigation of Hawaii — check out **liquidr.com/missions**.



The Wave Glider UMV can operate continuously over time periods of more than a year on power scavenged from ocean waves and sunlight.

Second-Generation Squirter

Proving that an advanced engineering degree from MIT doesn't guarantee that you will live a useful and productive life, graduates Bill Fienup and Barry Kudrowitz have come up with the AutoMato57 robotic Heinz ketchup dispenser which features the ability to maneuver to a plate and squirt the condiment onto a waiting hot dog or hamburger. This is actually a second-generation machine, being a "less anthropomorphic and multi-actuated version" of their earlier "Catsup Crapper" which traveled on roller skates instead of wheels or caterpillar tracks. Does it really cut the mustard? Decide for yourself. You can relish videos of both bots by logging onto **www.automato57.com**.

Shot from a video of the AutoMato57 robotic ketchup dispenser.

The Mega Hurtz SWAT bot from Inspector Bots.

Mean, But Not Lethal

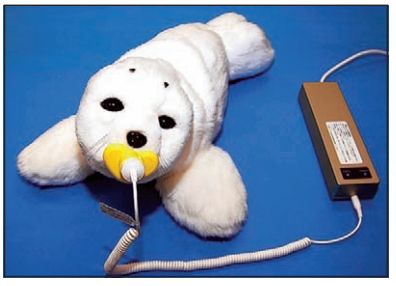
Considerably less huggable is Mega Hurtz, part of the family of tactical, inspection, and surveillance robots from Inspector Bots. This remotely operated four-wheel drive electric vehicle rolls along at speeds up to 7.5 mph, is powerful enough to tow a Hummer, and is touted as applicable to covert surveillance, security, SWAT, tactical response, and law enforcement duties. The mean-looking bot can be fitted with a variety of accessories, including a low-lux camera that provides night vision up to 35 ft in total darkness. Note, however, that Mega Hurtz is not as dangerous as it looks. The bot is strictly nonlethal, as the 20-round-per-second weapon mounted on its pan/tilt turret is actually a modified paintball gun with a 100-round magazine. It can also fire pepperballs or hardened rubber bullets to repel intruders, but it isn't likely to satisfy your blood lust. Curiously, we couldn't find the company's physical address on the Inspector Bots website, but the 303 area code would indicate that it's located somewhere in Colorado. For videos and more information, visit **www.inspectorbots.com**.

You Can't Hide From A Cougar

TiaLinx's Cougar10-L can see through reinforced concrete.

Another all-terrain bot designed for surveillance duty is the Cougar10-L from TiaLinx, Inc. (**www.tialinx.com**). This mini-robot sports a retractable arm that can be fitted with a variety of TiaLinx ultra-wideband, multi-GigaHertz RF sensors to perform the dual functions of through-the-wall imaging and underground unexploded ordnance (UXO) and cavity detection while humans remain at a safe stand-off distance. Multiple integrated cameras allow both day and night surveillance of a premise. The Cougar transmits directional wideband signals that can even penetrate a reinforced concrete wall, and a signal detector circuit captures reflections from targets, processes the signals, and provides remote wired or wireless real-time imaging. According to the company, it can scan and image concealed objects both vertically and horizontally, giving it more functionality than other systems. Ongoing modifications are aimed at extending its sensory capabilities to multi-story buildings.

Paro — a $6,000 cuddle bot for the elderly.



Baby Seal Therapy

Decades ago, gramps was a zealous anarcho-primitivist who spent his days saving whales and hugging sycamores. But now that he's confined to a nursing home and can't rove the primeval world anymore, how can he maintain a spiritual link with Mother Nature? Maybe the answer is Paro, a robotic baby harp seal designed to provide animal therapy in hospitals and extended care facilities where real live animals "present treatment or logistical difficulties." Paro was actually invented several years ago by Japan's National Institute of Advanced Industrial Science and Technology (AIST, **www.aist.go.jp**) and is in its eighth design generation. It reportedly never really caught on in Japan, so its creators obtained FDA Class II medical device clearance for it and are hoping it will fare better in the US. AIST notes that Paro has five types of sensors (tactile, light, auditory, temperature, and posture), giving him a wide variety of learning and response capabilities. However, one disappointed day care operator who gave Paro a try lamented, "It doesn't do much other than utter weird sounds like 'heeee' or 'huuuu' which probably won't go over with English speakers any better. In addition, you'll need to shell out about $6,000 for one, so maybe getting gramps a few potted plants would be a more sensible approach.

Classic Bot Show Available

We recently stumbled across one of the earliest TV appearances of a robotic character, an amusing 1953 episode of the *Adventures of Superman* entitled "The Runaway Robot." The plot involved an electronically controlled bot that falls into the hands of crooks who send it out to loot the First National Bank. Obviously, the Man of Steel is needed to stop the out-of-control bank robber. You can buy the entire first season (26 episodes) on DVD for about $12 from Amazon, but we located a public domain version in .avi format. You can download it at **www.jkeckert.com/freedownloads/supes_tv_0117.avi**. It's good for a few laughs. SV

George Reeves, a.k.a., Superman, with the Runaway Robot.

GEEKHEAD

by David Geer

Contact the author at geercom@windstream.net

DARwIn-OP Robot

Virginia Tech has a long history of robotics research including the DARwIn (Dynamic Anthropomorphic Robot with Intelligence) series humanoid robot platform which began in 2004 at the Robotics and Mechanisms Lab (RoMeLa) there to study bipedal locomotion, gait generation, and motion control. According to Dennis Hong, PhD, Director of RoMeLa, there have been six iterations of the robot. The first — DARwIn 0 — was a feasibility study prototype, constructed to determine what type of actuators to use and how to control them. RoMeLa's software made the first robot stand up and walk. However, without feedback data for stabilization retrieved from rate gyros and force sensors, the first robot fell over frequently.

Dennis Hong, Ph.D., Associate Professor, Mechanical Engineering Department, Director of Robotics & Mechanisms Laboratory, Virginia Tech with several of the DARwIn-OP robots.

The second DARwIn — DARwIn 1 — used simulations to help generate stable motion without the robot falling over. Researchers used kinematics and dynamic models of the robot together with easy-to-use GUIs to simulate gaits. By simulating gaits in software, the researchers were able to advance the robot's walking capabilities much more quickly.

In DARwIn 2, the researchers added true intelligence, as well as machine vision with IEEE 1394 cameras, an on-board computer that is a PC104+ running LabVIEW real time, kinematically spherical joints in the hips and shoulders, and lithium-polymer batteries.

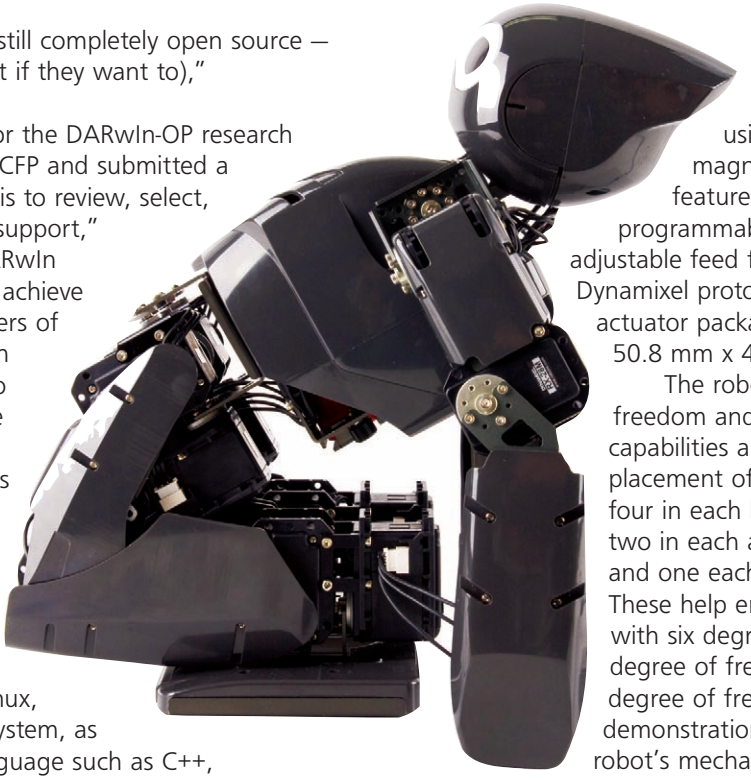
The robot can track objects with its cameras on its movable head, while the computer vision software locates and identifies objects based on camera input. With the combination of all these capabilities, DARwIn 2 is able to kick a ball autonomously into a goal which helped it to become the first humanoid robot in the US to be qualified for the international autonomous robot soccer competition RoboCup in the humanoid division.

Due to demand, Virginia Tech in conjunction with robotics labs at the University of Pennsylvania and Purdue University, and the Korean robot company Robotis is producing a fully open source hardware and software robot platform called DARwIn-OP for education outreach and research, with support from the NSF (National Science Foundation).

"We have been using Robotis' Dynamixel servo actuators for the DARwIn series humanoids since 2004 and are very happy with them. Thus, it is natural that Robotis would be a partner for this project. But note that though

Robotis will be selling it, it is still completely open source — anyone can build it (and sell it if they want to)," commented Hong.

The NSF is the sponsor for the DARwIn-OP research project. "We responded to a CFP and submitted a research proposal. NSF's role is to review, select, and fund projects worthy of support," explained Hong. With the DARwIn platform so well equipped to achieve the goals of the NSF and others of increasing the number of high school students who go on to become college and graduate school students in science, engineering, and mathematics programs, it was only natural that the NSF would want to fund this research project to help in achieving those goals.



The robot is PC based and researchers can install Linux, Windows, or any operating system, as well as any programming language such as C++, LabVIEW, or MATLAB, according to Hong.

Because it is an open source research platform, the CAD files, assembly manuals, and fabrication manuals for the hardware will be available free on-line. This means anybody who has sufficient skills and equipment (and budget) will be able to make one themselves. The DARwIn-OP team also encourages users to modify it and hack it, and share it on-line to form a user community. The robot has sufficient torque for various research scenarios and enough payload to add different sensors, computing platforms, and components.

Potential payloads for research include adding LIDAR, a stereo camera, a dexterous hand, and additional computational/signal processing equipment. For experiments, it may pick up objects like cups or balls, and perform cooperative manipulation of boxes, according to Hong.

Starting With the Right Actuators

Crucial to DARwIn-OP's motor skills are its 20 Dynamixel RX-28M actuators which actuate its arms, legs, body, and head. The RX-28M is a new actuator developed for DARwIn-OP by Robotis. The actuators each employ one Maxon RE-Max customized DC motor. The gear reduction ratio is 193:1.

The actuator comes with up to 4.5 Mbps buffered TTL interface speed and up to 4,096 resolution feedback using an accurate non-contact magnetic encoder. Additional features include user programmable PID gain, auto-adjustable feed forward control, and Dynamixel protocols 1.0 and 2.0. The actuator package measures 35.5 mm x 50.8 mm x 41.8 mm.

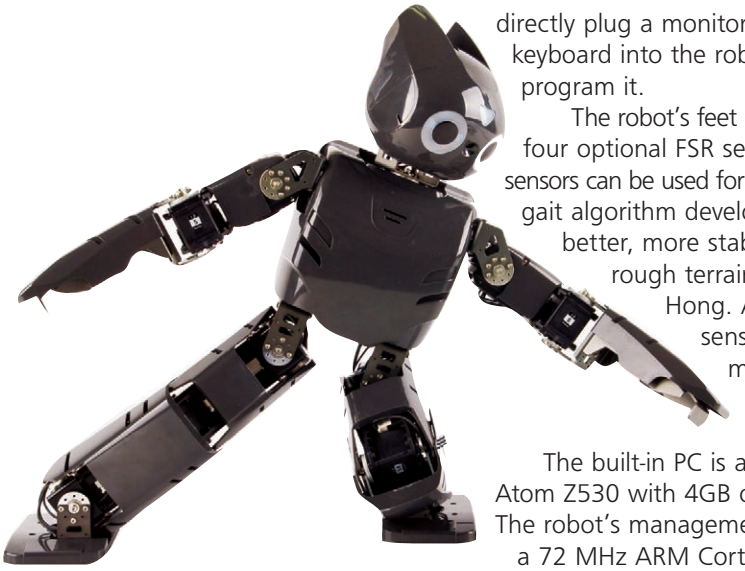
The robot's many degrees of freedom and overall gymnastic capabilities are supported by the key placement of these actuators, including four in each leg, four in the lower body, two in each arm, one in each shoulder, and one each for the head and neck. These help enable the robot platform with six degree of freedom legs, three degree of freedom arms, and a two degree of freedom neck. The gymnastic demonstrations that result show off the robot's mechanical abilities nicely.

The robot can easily stand from a face down or face up position, walk, bend over backwards, and do head and hand stands. It has the capability to even run with two feet off the ground. With additional programming, components and hardware that researchers are free to add, the robot can learn many new tasks and capabilities, as well.

Overall Features

The robot's features include a high definition USB camera head mounted for computer vision, eye and forehead based LED lights to display the robot's status, a USB mic for audio inputs, as well as two optional microphones — one mounted on each side of the head. "We use the status LEDs to show the status, mode, and state of the robot. The LEDs in the eyes could also be used to show emotion for HRI which is important for that type of work," stated Hong.

The robot can communicate via a chest based speaker. The robot maintains its balance and coordinates using a three-axis gyroscope and three-axis accelerometer. The robot's torso is equipped with a mini SD card and slot, Wi-Fi, two cooling fans, two USB interfaces, HDMI port and capability, audio line-in and line-out ports, an Ethernet port, an external power source, and a removable handle. Researchers can



directly plug a monitor and a keyboard into the robot to control or program it.

The robot's feet can come with four optional FSR sensors. "These sensors can be used for ZMP feedback, gait algorithm development, and better, more stable walking in rough terrains," explained Hong. Additional sensors include a magnetic encoder for each joint.

The built-in PC is a 1.6 GHz Intel Atom Z530 with 4GB of SSD flash. The robot's management controller is a 72 MHz ARM CortexM3 STM32F103RE. DARwIn-OP comes with a kill switch in its back.

The robot arms come in the basic model with three DOF and no real hand gripping, the gripper arm model with four DOF and the ability to grip objects, and the yaw joint appended gripper arm with five DOF.

Conclusion

Researchers from various educational institutions have already applied DARwIn-OP to many new research directions including vision processing, mechatronics, machine design, autonomous behavior, multi-robot control, dynamics and control, simulation, kinematics, motion generation, manipulation, and human robot interaction, according to Hong.

This robot is a research platform that comes with software and everything is open source, but its capability depends on the user's programming. "UPenn's GRASP lab has been using DARwIn for machine learning algorithms to make it do many things such as recovering from a push while walking," commented Hong. SV

Resources

Virginia Tech
http://www.vt.edu

DARwIn-OP project on SourceForge
http://sourceforge.net/projects/darwinop

DARwIn-OP movie
http://voxel.dl.sourceforge.net/project/darwinop/etc/Movie/DARwIn-OP_Whole.wmv

DARwIn-OP at 2010 IEEE-RAS International Conference on Humanoid Robots
http://bit.ly/fGzwvq

DARwIn-OP movie 2
http://bit.ly/gS0aEG

Lynxmotion

The Lynxmotion Servo Erector Set
Imagine it... Build it... Control it!

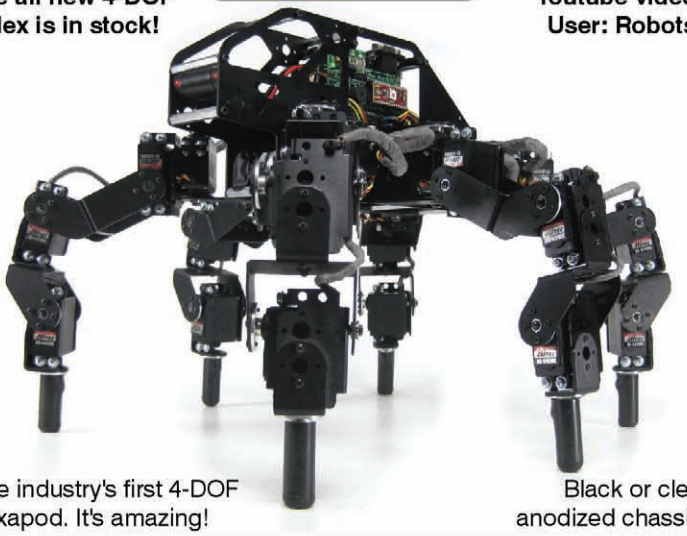
Biped Nick

Biped Pete

Featured Robot

The all new 4-DOF T-Hex is in stock!

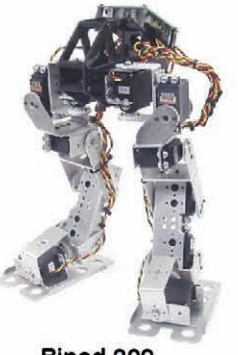
Youtube videos User: Robots7



The industry's first 4-DOF hexapod. It's amazing!

Black or clear anodized chassis!

Biped Scout



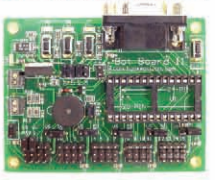
Biped 209

Walking Stick

With our popular Servo Erector Set you can easily build and control the robot of your dreams!

Our interchangeable aluminum brackets, hubs, and tubing make the ultimate in precision mechanical assemblies possible.

All New ARC-32 - $99.95
32 Channel Servo/Microcontroller.
USB Programming port.
32 bit Hardware based math.
Program in BASIC, C, or ASM
Servo and Logic power inputs.
Sony PS2 game controller port.



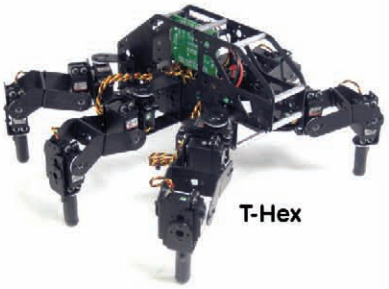
Bot Board II - $24.95
Carrier for Atom / Pro, BS2, etc.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
Buffered Speaker.
Sony PS2 game controller port.
3 Push button / LED interface.

SSC-32 - $39.95
32 Channel Servo Controller.
Speed, Timed, or Group moves.
Servo and Logic power inputs.
5vdc 250mA LDO Regulator.
TTL or RS-232 Serial Comms.
No better SSC value anywhere!

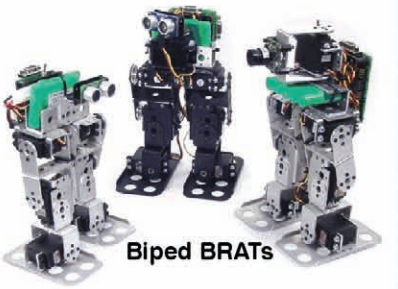
We also carry motors, wheels, hubs, batteries, chargers, servos, sensors, RC radios, pillow blocks, hardware, etc!

Visit our huge website to see our complete line of robots, electronics, and mechanical components.

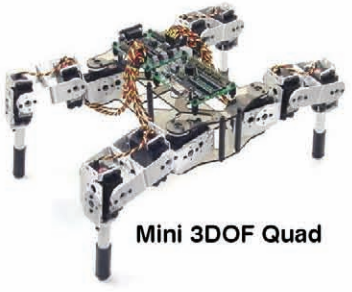
CH3-R



T-Hex



Biped BRATs



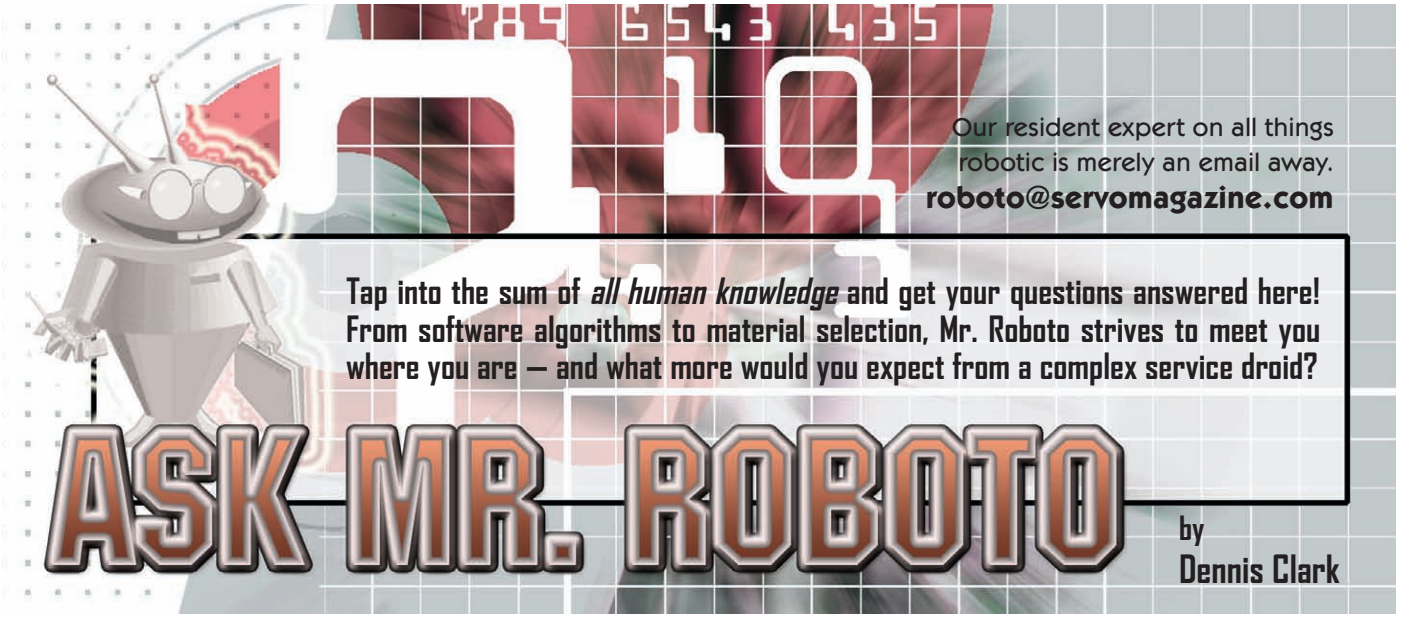
Mini 3DOF Quad

Phoenix

Images represent a fraction of what can be made!

www.lynxmotion.com

The SES now has over 200 unique components!



Our resident expert on all things robotic is merely an email away.
roboto@servomagazine.com

Tap into the sum of *all human knowledge* and get your questions answered here! From software algorithms to material selection, Mr. Roboto strives to meet you where you are — and what more would you expect from a complex service droid?

ASK MR. ROBOTO

by
Dennis Clark

Q. I am building a robot to destroy tanks and take over the world. My largest problem is getting the solder to stick to the wires. If I could just solve that, I'd be all set. Do you have any suggestions?

— Anonymous

A. While I'm not a big fan of military robots, even I can see the utility of a robot that can take over the world. Soldering is your first step into the world of electronics and can be frustrating. I suggest that you get a Weller pistol-grip soldering iron, plug it in and then open your hand and place the glowing tip ...

April Fool! Okay, that lame example is about as much as you'll get from me on the traditional April Fool's jokes. Now on to the fun stuff. I have a good variety of questions this month; folks must be getting ready for exams or perhaps the summer spate of robot competitions. Let's get started.

Q. I am embarrassed to ask, but I need your help. I do a little programming with the PIC16F84A and PIC16F877A. I am having a big problem to come up with the code for my project which uses two servos. I'm using a 4 MHz crystal which is divided by four to give me 1 MHZ, then I'm using the TMR0 counter to divide by 256 which gives me 3906 Hz. What I need to do is come up with three delays (1) 1 ms, (2) 1.5 ms, and (3) 2 ms at a rate of 20 ms period. I have spent a lot of time trying to accomplish this to no avail. I am using assembly code for this project. I have looked on the Internet and all I get is code in C or PICBasic. If you can help, I sure would appreciate it. I am a tech not a programmer, even though I have done a few things using this code, just not with timing pulses. Thanks,

— Frank G.

A. It sounds like you only need to have three distinct servo pulses for your project. It really isn't that much more difficult to have every possible servo pulse with an eight- bit resolution in your project. I've done very similar assembler code in a PIC12F508 part. Let's look at some code to show you some of the tools you'll want to know for timing pulse widths.

The first thing you'll need to do is set up your timer. There are two things that you need to time: the first is the span between the pulses which is 20 mS, the second is the actual pulse width. The canonical range of pulses for a hobby servo is 1 mS to 2 mS for their full span. In reality, most servos will properly respond to pulses a little shorter and a little longer. Many that I've played with will handle 0.8 mS to 1.2 mS with no trouble. Test yours before you count on that little bit of trivia!

First, let's decide what we need for timing. We'll start with the pulse width since that is *far* more critical than the span between pulses (more on that later.) We want — at the minimum — to be able to time from 1 mS to 2 mS with reasonable resolution. So, with this in mind, let's find out what our eight-bit timer should have for each "tic."

```
1mS/256 = 3.9uS
```

This means that with a 3.9 µS timer clock, a full count on (TMR0, for instance) will yield exactly 1 mS time elapsed. Different PICs have different ways to set timer prescales and counts though; so I'll pick your 16F84 part (you should step up to newer PICs though; they are easier to work with). We never will get an exact divider from this PIC, however. This type of PIC processor has a four stage pipeline which means that the actual clock that the TMR0 will get is 1 MHz. This is a 1 µS clock, so we will have to prescale our clock as a divide-by-4 to get a 4 µS clock to the timer. This will yield a 1.024 mS full range. Close enough. The following code will set TMR0 up to give us this useful timer range.

```
movlw   0x01            ; Set clock prescale Timer0
                        ; to divide by 4
movwf   OPTION_REG      ; On your part this may
                        ; just be "OPTION"
```

I'm going to make this primitive to get my point across; you can make it more sophisticated as you develop your program. Let's assume that you have one of your servos set up on PORTB bit 0. I'll show a simple code sequence that will set the pulse to whatever you have in memory location PULSE. When checking our timer, we check for it to reach the 0 count. So, if we want to count to 200, we set the timer value to be 255-200 so that after 200 counts, we are at 0. This is easy to do in assembler:

```
Servo_loop
  bcf   PORTB,0  ; clear the servo pulse to '0'
  movlw D'5'     ; 250 * 4uS = 1mS time waste.
  movwf TMR0     ; set the timer

Waste
  movf  TMR0,W   ; get the timer value
  btfss STATUS,Z ; check for the "zero" flag
  goto  Waste    ; nope, not zero

Pulse
  bsf   PORTB,0  ; set servo pulse high
  movf  PULSE,W  ; get our pulse (255-count
                 ; time remember)
  movwf TMR0     ; set the timer

GoHigh
  movf  TMR0,W
  bfss  STATUS,Z ; Are we there yet?
  goto  GoHigh   ; nope
  bcf   PORTB,0  ; yes, clear IO line to '0'
  movlw D'20'    ; set the counter at this
                 ; memory location
  movwf COUNTER  ; where we track times through
                 ; 1mS loop

Wait20ms
  movlw D'5'     ; 255-5=250, 250*4uS=1mS, do
                 ; 20 times

Wait1ms
  movf  TMR0,W   ; get timer value
  btfss STATUS,Z ; are we there yet?
  goto  Wait1ms  ; nope

  decf  COUNTER,F   ; yes, decrement our loop
                    ; count
  btfss STATUS,Z    ; are we to 20mS yet?
  goto  Wait20ms    ; nope
  goto  Servo_loop  ; yes! Keep doing the
                    ; servo pulses
```

This code will generate any servo pulse from 1 mS to 2 mS with a 4 µS resolution. If you have a more sophisticated PIC, you can set TMR0 to cause an interrupt when the rollover to zero occurs, and in your ISR (Interrupt Service Routine), you can set the timers for the next state you want to time. In this way, the servo timing will take place "in the background" and your program can be doing other things while your ISR happily takes care of the tedious servo timing details.

Q. Dear Mr. Roboto, I love all things robot, servo, and PIC related. It is with some disappointment that I have not been able to find a good article anywhere on how to build a PIC controlled 16 servo controller. Now, I can find plenty of kits or pre-built products. I can also find plenty of articles on how to build an eight servo controller. What I would like to see is how to DIY a 16 servo controller, controlled by a PIC, built from scratch. Can you point me in the right direction? As an alternative, is there any sense in just combining two eight servo controllers together with a PIC brain controlling both? One question that arises from the last idea is wouldn't that throw a kink in walking gaits and walking modes? Perhaps preventing the robot from multitasking? Thanks for your help.

— Jason Travis

A. The reason that you don't find more resources for more than an eight servo controller is that it gets lots more difficult once you pass that plateau. The higher count servo controllers are then written by folks that want their efforts paid for — at least that's my guess. The rest of us are perfectly happy to pay for that work since we gave up writing our own. I myself have written four servo controllers just for a cheap way to drive little servo-based robots and other inexpensive projects. As soon as I needed better resolution, more servos, and control of transit rates, I bought specialized controllers.

There are several companies supplying reasonably priced servo controllers as you have discovered. I recommend that you go that route. I know, it offends the DIY in all of us to buy rather than build, but what is your end goal? To make a servo controller or to make a robot *with* the servo controller? To answer your question, yes, by spanning two controllers you will lose control of the sequencing in your walking gaits if you are synchronizing them through your controller. If you are doing your sync timing in your main program and just sending servo commands out individually, then I don't think it will matter.

Q. I want to use a laptop with one or more USB joysticks to control my underwater ROV. I have no idea, however, how to code the actual reading of the joystick axes and buttons. Once I can get the raw data into a C program (preferably using gcc and public libraries), I can write the code to convert axis movement in forward, reverse, rotate, up, down, arm move, ballast, etc.

Can you point me to the interface library and a sample interface code?

— Mike

A. USB joysticks are HID devices and as such can be accessed the same way you can write programs to get data from a mouse or keyboard. You can get access to the Windows HID API at **http://msdn.microsoft.com/en-us/library/aa973512.aspx**. Apparently, Microsoft also has information on using their HID driver API at **www.microsoft.com/whdc/devtools/WDK/default.mspx**.

This is a bit of a *heavy lift* for the casual user, though.

I looked for quite a while and did not find anything helpful as demo code to use this. A good look through the API manual should give you what you need. Once you understand the USB HID API and have a program running, it will be a simple matter to send commands out to your robot. Because this *is* a bit arcane, I recommend a book. When dealing with things USB, the place to go is Jan Axelson's *USB Complete*.

Follow-up to a February 2011 Question

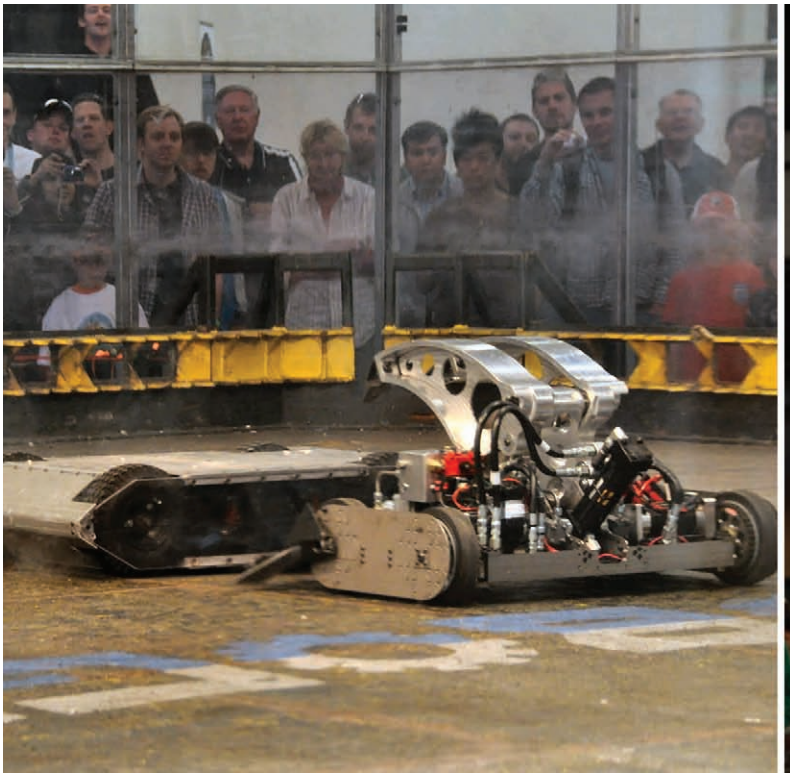
In the February issue, I was asked about controlling an RC aircraft transmitter from a computer. I gave the basics for a DIY solution that required a lot of work to finish. Just recently, I came across a company that has done the work for you! Check out Mile High Wings at **http://milehighwings.com** and look into their USB v.4 Interface.

Well, we've come to the end of another Mr. Roboto and as always, if you have a question, drop me a line at roboto@servomagazine.com. I'll be happy to try to answer it. Have fun and keep building robots! **SV**

www.servomagazine.com/index.php?/magazine/article/april2011_MrRoboto

ROBOGAMES

"World's Largest Robot Competition" - *Guinness Book of Records*
"Top 10 Video Highlights" - ESPN *SportsCenter's Play of the Day*
"The best robots compete in RoboGames, just as the best athletes train for the Olympics" - *Discover*
"The Best Ten North American Geek Fests" - *Wired*



RoboGames is the olympics for robots – a three-day event in the San Francisco Bay Area that brings the smartest humans and best robots from around the world to compete in a wide array of robotics oriented events (over 40 countries have participated in past events.)

Cart-wheeling androids, combat robots, autonomous vehicles, LEGO robots, soccer bots and even cocktail-mixing barbots – there's something for everyone at RoboGames! The event also features demonstrations by leading robotics industry designers and engineers, kinetic art exhibits, and the latest in tech products, gear, and innovation.

RoboGames began as an enthusiastic experiment in robotic cross-pollination, when dozens of disparate, well-established robot competitors were placed under the same roof. Bringing together builders from acclaimed fields such as combat robotics, robot soccer, sumo, fire-fighting, androids, and kinetic art, RoboGames enables robot builders to exchange ideas and share their knowledge and experience with each other. Varying disciplines now learn from one-another and the event has grown into a fantastic multi-layered, multi-cultural experience like no other. The best part is that RoboGames is completely open–anyone can compete: competitors have been garage builders, K-12 school teams, professional engineers, and university researchers. Come see the future evolve!

Educational Outreach: In addition to the 60 adult events, RoboGames sponsors 10 different "junior league" events for kids in K-12, which are free for kids to compete. University students also have the opportunity to publish and present research papers.

Sponsors: Your company can reach millions of people around the world by sponsoring RoboGames. Tens of thousands of people attend the event in person, and millions are reached from the media-exposure – print, web, radio, and television. Popular with techies, sports-fans and hipsters alike, RoboGames has something to offer every demographic.

Go on-line to find out more! Watch videos, get building tips, register to compete, buy tickets or sponsor the next event.

This years' event will be taped by The Discovery Network for a TV show to be released this year! If you can't compete or attend, be sure to watch for us on TV!

April 15-17th, 2011 - San Mateo, California - http://RoboGames.net

EVENTS Calendar

ROBOTS.NET

Send updates, new listings, corrections, complaints, and suggestions to: steve@ncc.com or FAX 972-404-0269

Know of any robot competitions I've missed? Is your local school or robot group planning a contest? Send an email to steve@ncc.com and tell me about it. Be sure to include the date and location of your contest. If you have a website with contest info, send along the URL as well, so we can tell everyone else about it.

For last-minute updates and changes, you can always find the most recent version of the Robot Competition FAQ at Robots.net: **http://robots.net/rcfaq.html**

— R. Steven Rainwater

APRIL

2 **CIRC Central Illinois Bot Brawl**
Lakeview Museum, Peoria, IL
Lots of events include 3 kg Sumo, 500 g Sumo, LEGO Sumo, line following, maze following for autonomous bots, and several combat events for remote control.
http://circ.mtco.com/competitions/2011

2-3 **Trenton Computer Festival Robotics Contest**
College of New Jersey, Ewing Township, NJ
Not sure what the specifics are yet, but based on last year's event, they'll have a variety of robot demonstrations and contests.
www.tcf-nj.org

9-10 **Trinity College Fire Fighting Home Robot Contest**
Trinity College, Hartford, CT
Autonomous robots must navigate through a mock house, then locate and extinguish a candle in the shortest time possible.
www.trincoll.edu/events/robot

10 **Robotics Innovations Competition and Conference**
Woburn, MA
University students must engineer robots to solve real world problems. This year, they must demonstrate robots that use unconventional means of mobility to navigate through environments that are inaccessible to wheeled and tracked robots.
http://ricc.wpi.edu

14-16 **National Robotics Challenge**
Marion, OH
A variety of student robot events including Robo Hockey and Sumo.
www.nationalroboticschallenge.org

14-16 **VEX Robotics World Championship**
Kissimmee, FL
High school and university VEX teams compete with robots that are operated in both autonomous and remote control modes.
www.vexrobotics.com/competition

15 **Carnegie Mellon Mobot Races**
CMU, Pittsburgh, PA
The famous annual CMU line following autonomous robot races.
www.cs.smu.edu/~mobot

15-17 **RoboGames**
Ft. Mason's Festival Pavillion, San Francisco, CA
FIRA, BEAM, Mindstorms, and lots of other events for autonomous and remote-control robots.
www.robogames.net

16 **Greater Philadelphia Sea Perch Challenge**
Drexel University, Philadelphia, PA
Students compete with tethered underwater ROVs.
www.coe.drexel.edu/seaperch

16 **Penn State Abington Fire Fighting Robot Contest**
Penn State Abington, Abington, PA
Autonomous bots must navigate a maze and put out a fire.
www.ecsel.psu.edu/~avanzato/robots/contests

16 **Penn State Abington Mini Grand Challenge**
Penn State Abington, Abington, PA
Autonomous bots must navigate an outdoor course.
www.ecsel.psu.edu/~avanzato/robots/contests

16 **RoboRodentia**
California Polytechnic, San Luis Obispo, CA
Autonomous micromouse-like robots navigate a

maze, pick up balls, and place them into a nest.
https://sites.google.com/site/calpolycomputerengineering

23 **SparkFun Autonomous Vehicle Competition**
Boulder, CO
Autonomous ground and air robots must circumnavigate the SparkFun building.
www.sparkfun.com/products/10435

27-30 **FIRST Robotics Competition**
Edward Jones Dome, St. Louis, MO
Teams of high school students design robots to compete in a different competition each year.
www.usfirst.org

30 **Hawaii Underwater Robot Challenge**
Kahanamoku Pool, UoH at Manoa, Honolulu, HI
Students compete with tethered underwater ROVs in this regional for the MATE ROV competition.
www.marinetech.org/rov_competition

30 **National Electronics Museum Robot Festival**
National Electronics Museum, Linthicum, MD
Lots of robot events including Fire Fighting, FIRST, Sumo, robot combat, plus other Maker Faire-like activities and events.
www.robotfest.com

30 **The Tech Museum of Innovation's Annual Tech Challenge**
Parkside Hall, San Jose, CA
This year's challenge is called "Explore the Volcano."
http://techchallenge.thetech.org

NEW PRODUCTS

TOOLS & ACCESSORIES

X2 Ultima Two-Channel Charger

Recharging your high amperage batteries has never been easier than with Hitec's all new X2 Ultima two-channel charger. Featuring dual 200 watt power ports, this efficient charger will charge two lithium, NiCd, NiMH, or lead acid packs at once. It is capable of delivering 10 amps of charging current and can discharge your packs at up to five amps for efficient battery maintenance.

Perfect for electric aircraft and surface hobbyists, the X2 Ultima will power larger batteries in a short period of time. It includes two balancing boards and an assortment of connectors at a price of $159.49.

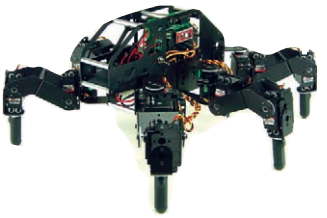
All Hitec's products carry a two year warranty against manufacturing defects from date of purchase.

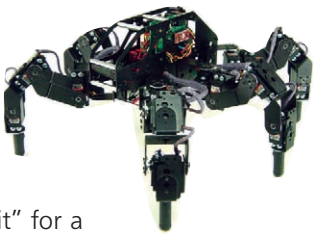
For further information, please contact:

Hitec
12115 Paine St.
Poway, CA 92064
858•748•6948 Fax: **858•748•1767**
Website: **www.hitecrcd.com**

KITS

Hexapod Walking Robot Kits





Lynxmotion announces the release of two new Hexapod walking robot kits. Heavily inspired by the work of robot guru Kåre Halvorsen, the 3-DOF T-Hex and the 4-DOF T-Hex designs differ from previous offerings in several ways.

The "mech" inspired chassis uses a DVP (dual vertical panel) design. This allows more flexibility when placing electronics and batteries. It also has a "cockpit" for a pilot. The use of offset brackets on the legs provides for more range of movement. The 4-DOF T-Hex is an industry first as the only commercially available 4-DOF (degree of freedom) hexapod robot.

The robot has a massive amount of ground clearance, making it able to walk right over other servo-based robots. The programming includes PS2, XBee, and serial control options. There is also a new insect-like "triple tripod" walking gait. The 3-DOF T-Hex offers the builder a less expensive version. While very capable on its own, it can be upgraded to the 4-DOF version at any time with additional servos and SES (Servo Erector Set) parts. The leg design uses the offset brackets for increased range of movement.

Planned accessories include a micro servo pan and tilt, an integrated turret, and a vacuum molded body. In stock and ready to rock, T-Hex is the ultimate in mech inspired hexapod nirvana. Check out the website listed for videos and images.

For further information, please contact:

Lynxmotion Website: **www.lynxmotion.com**

Robot Base Kit

Parallax is now offering a robot base kit that is available in either white or black high-density polyethylene (HDPE). Ideal for medium-sized autonomous robotics system development, this sturdy platform can handle indoor and outdoor environments.

The kit includes the base plate, acrylic battery shelf, and a set of mounting hardware for attaching Parallax's 12V motor mount and wheel kit, and two caster wheel kits; these kits are sold separately.

Features:

- Precision machined HDPE base plate with pre-drilled mounting holes.
- Laser cut 1/4" thick acrylic battery shelf.
- Mounting hardware 7/64" hex key included for quick assembly.
- Designed and pre-drilled for attaching other kits.
- Pre-drilled for use with up to 10 PING))) ultrasonic

distance sensors and protector stands (sold separately)
- HDPE base plate is virtually indestructible, yet easily drilled, cut, or modified to suit design preferences and application needs.
- 0.375" (0.95 cm) thick x 17.75" (45.09 cm) diameter HDPE base plate.
- 0.250" (0.64 cm) thick x 4.5" x 13.25" (11.43 x 33.66 cm) acrylic battery shelf.
- Sturdy but not heavy; 3.15 lbs (1.43 kg).

Visit the website listed and search "Robot Base Kit." Price is $39.99.

For further information, please contact:

Parallax, Inc. Website: **www.parallax.com**

INDUSTRIAL

110 vAC Power Solution With c3 Compact Six Axis Robots

EPSON Robots introduces a high performance 110 VAC power solution with EPSON C3 compact six axis robots for non-standard manufacturing environments.

This power solution is available with Epson's true PC based RC620+ controller which provides the power and flexibility of an open architecture solution that EPSON PC based controls are known for. The RC620+ controller offers ease of use with their RC+ controls software and lots of fully integrated options such as: vision guidance, .Net support, Profibus, DeviceNet, EtherNet/IP, and more.

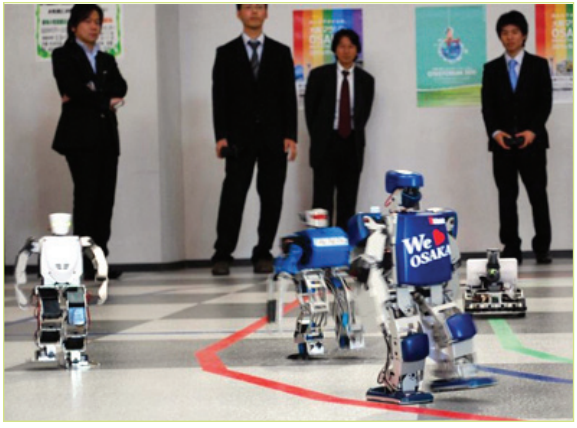
For further information, please contact:

EPSON Robots
18300 Central Ave.
Carson, CA 90746
Tel: **562•290•5910**
Website: **www.robots.epson.com**

Is your product innovative, less expensive, more functional, or just plain cool? If you have a new product that you would like us to run in our *New Products* section, please email a short description (300-500 words) and a photo of your product to:

newproducts@servomagazine.com

bots IN BRIEF



MARATHON MEN

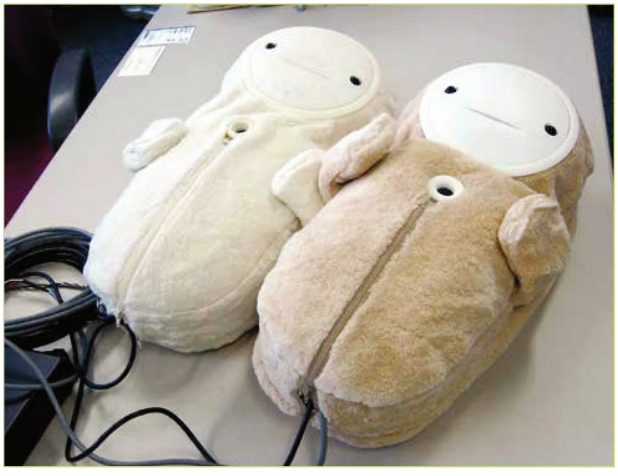
This last February, the first robot marathon was held in western Japan. Organized by robot technology firm Vstone Co., in cooperation with Osaka, five two-legged bots had to complete 422 laps around a 110 yard indoor track. (That equals 26 miles.) The "Robo Mara Full" race winner completed his walk in 54 hours, 57 minutes, and 50.26 seconds. The group is hoping that the event will go worldwide yearly.

- 1st place: 54 h 57 m 50 s for Team Vstone's autonomous Robovie-PC
- 2nd place: 54 h 57 m 51 s for Center Team's silver Robovie-PC

The Osaka Institute of Technology's Team A stalled after 33.795 km and the Job Creation Team made it 17.723 km. The Osaka Institute of Technology's Team B failed after only 0.224 km.

TO THE MOON

The Google-promoted Lunar X-Prize has been narrowed down to 29 teams from 17 countries. The winner of the $30 million will get to send their low budget robot to the Moon, where it will send back photos. The competition was first announced in 2007 and has a target date of completion by 2015. The teams vary in make-up from university groups to commercial businesses. Some of them have already contacted spacecraft companies to transport their bots.



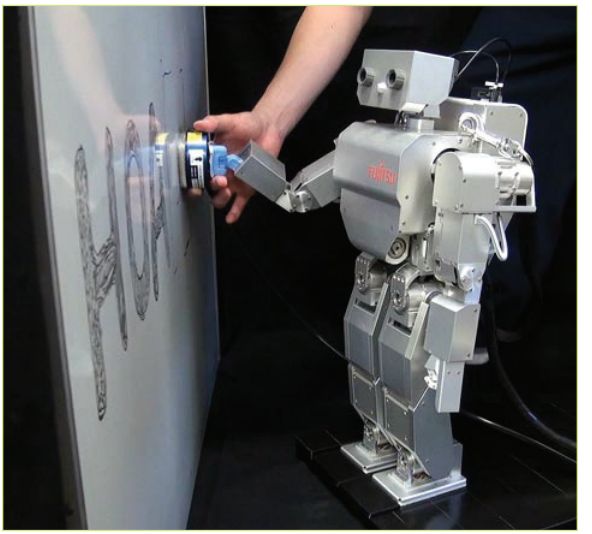
BABY ME

A new therapeutic robot called the Babyloid has been developed that may help relieve symptoms of depression in the elderly. The robot is a natural extension of baby doll therapy, encouraging the patient to take on an active care-giving role. Its unusual appearance treads a line somewhere between a mechanical object and a living being, with the general design motif inspired by a Beluga whale calf. It measures approximately 44 cm in length and weighs 2.2 kg. The robot's arms can wiggle, its eyelids can blink, and its mouth can open thanks to built-in actuators. The silicone resin face is loaded with LEDs to create the illusion of running tears or red cheeks.

The robot was developed by Kanou Masayoshi, Associate Professor at Chukyo University of Science and Technology, who has worked with Business Design Laboratory on the ifbot (communication robot) and Mechadroid Type C3 (receptionist robot). Essentially, the robot can simulate a bad mood (such as hunger or crying) and then will settle down.

Experiments were conducted at a nursing home over the course of two weeks with five subjects. Active observation and interviews each day assessed whether patients accepted the robot or not, and psychological changes before and after use such as any effects on mood. More experiments with the Babyloid are needed, and some 10 companies from the area will collaborate to build a new prototype over the next two years. Professor Kanou has stated he hopes such a robot could sell for around 50,000 JPY ($600 USD) in the future.

WIPEOUT



An Italian-Japanese collaboration between the Italian Institute of Technology and Tokyo City University has taught a small humanoid robot (Fujitsu's HOAP-2) how to wipe clean a common whiteboard. Led by Dr. Petar Kormushev (IIT) — who previously taught the iCub to use a bow and arrow — and Prof. Dragomir N. Nenchev (TCU), the project used kinesthetic teaching to inform the robot's movement.

A small eraser was attached to the robot's hand which was guided in a variety of sweeping patterns by a human instructor. A force-torque sensor attached to the robot's wrist recorded the patterns and forces applied to its hand which the robot would then replicate. The lower body was controlled separately by an ankle-hip algorithm that helps the robot maintain its balance (developed at TCU's Robot Life Support Lab).

Although the researchers admit the robot is not using its stereoscopic vision to detect exactly where the whiteboard needs to be cleaned, that could be implemented in the future relatively easily. For now, they want to focus on teaching the robot new skills through hands-on instruction which could be used to teach robots a variety of motion tasks.

COMBAT ZONE

Featured This Month:

Features

Events

www.servomagazine.com/index.php?/magazine/article/april2011_CombatZone

BUILD REPORT

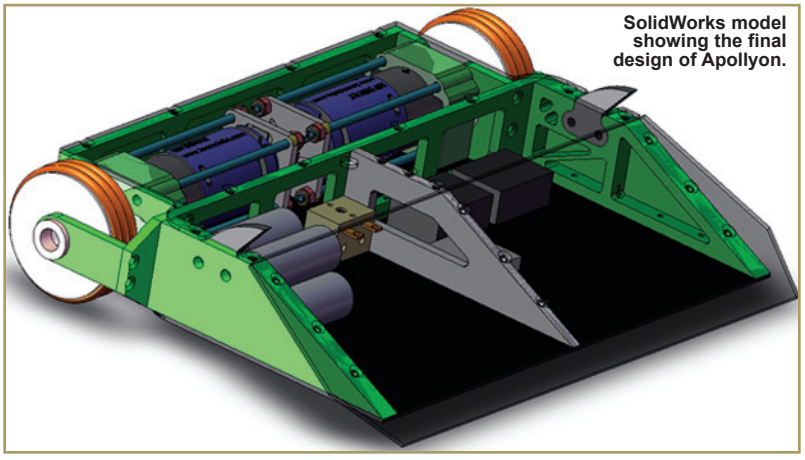
Apollyon 2 – Concept to Creation

● by Mike Jeffries

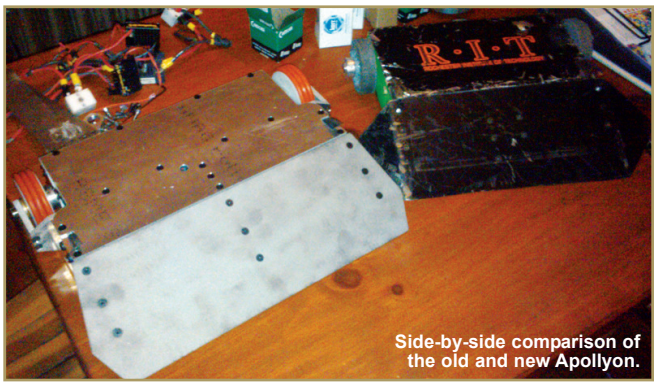
When this design was last discussed, it had been decided that the drive system would be switched from the discontinued GB-42 gearboxes to the widely available, heavier, and more powerful 18V Dewalt Power Drive Kit from Team Delta. Since then, there have been some refinements and tweaks to the design to optimize it and streamline use.

The battery pack has been switched from a 5s1p A123 Systems pack to a 6s1p pack, to increase voltage and maximum power. This pack coupled with the 18V Dewalt motors results in 2 HP peak output in a 12 lb robot. The change in gearboxes and motors results in no change in top speed, but it is able to attain maximum speed in a much shorter distance.

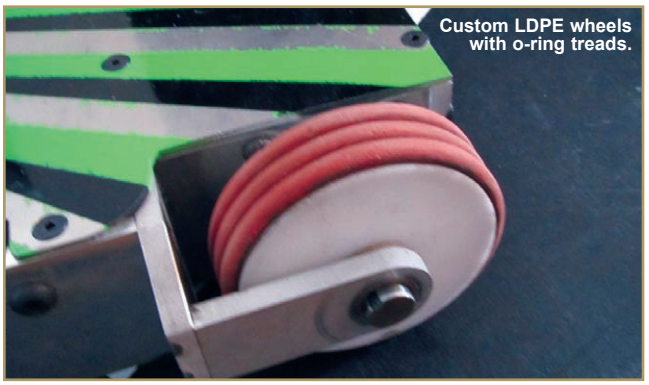
The new wheels were designed for easy maintenance, low cost replacement, and higher traction. The main hub is a keyed

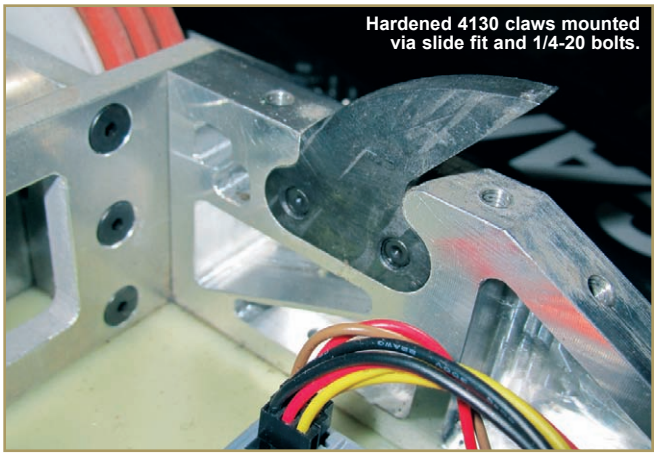


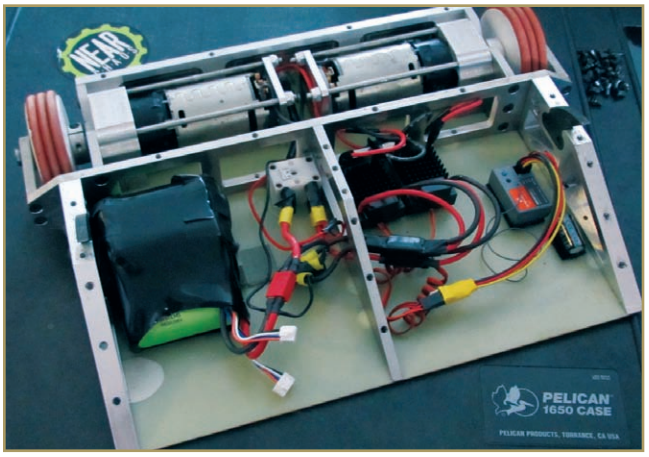
SolidWorks model showing the final design of Apollyon.



Side-by-side comparison of the old and new Apollyon.


Custom LDPE wheels with o-ring treads.


Hardened 4130 claws mounted via slide fit and 1/4-20 bolts.



Internal shot of Apollyon showing new electronics and layout.

LDPE disk with three o-ring grooves turned into it. The o-rings are 2-7/8" OD, 1/4" cross section silicone. The use of three o-rings as the tread material should allow it to more easily deal with debris on the arena floor.

The claws mounted to the top of Apollyon are now much stronger. In addition to being made of 4130 steel that has been heat treated to 45 HRC, they are now fitted into the frame rails on the side of the chassis. This greatly increases the mounting strength and will result in them being much harder to remove during an impact. The bolt holes in the claws are threaded to allow them to be used as their own nuts which will reduce the weight slightly over a through-hole with a nut on the end.

All of these changes — most of which resulted in the addition of weight — caused a need for lighter electronics. This meant the old, bulky Victor 883 speed controllers were out. I searched for a while before discovering the controllers that I decided to use. According to the manufacturer, they'll handle 80A continuous and will take up to a 6s lipoly/A123 pack. After some testing, they seem to fit the bill perfectly and at only 38 g each, they're a major weight savings over the Victor 883's used in previous versions of Apollyon. A review of the Holmes Hobbies' BR-XL speed controllers will be coming in the next few months.

Apollyon in fully combat-ready state.

Much of the machining was done by Team Whyachi and as a result, construction went quickly. As of now, Apollyon is 100% combat ready and by the time this is published, will have competed at Motorama in Harrisburg, PA. SV

SO YOU WANT TO FIGHT ROBOTS? Turning Your Destructive Instincts into a Socially Acceptable Hobby!

● by Kevin Berry

Mostly, when the general public hears that someone's hobby is "robotics" they think of a fragile collection of wires, struts, motors, and glowing eye-like things. The more tech savvy may think of VEX, LEGO Mindstorms, or even Robonova. Speak of a robotics competition, and they might come up with FIRST, or maybe even the DARPA Grand Challenge.

If you are a regular reader of The Combat Zone, however, you know there's a dark side to the hobby. It is a small but dedicated, continent spanning band of tinkerers with a perverse interest in spending months of their spare time and mounds of money to craft a nasty looking, remote-controlled bot that is often destroyed in under a minute.

Photo by Sam Coniglio.

We're talking about fighting robots of course. It was introduced to the public on television through the Robot Wars, Robotica, and BattleBots series which was followed by a line of toys distributed through stores and as Happy Meal toys through McDonald's restaurants. The term "BattleBots" quickly became a generic name for fighting robots; a la "Kleenex" or "Xerox."

SERVO Magazine has always been a strong supporter of the sport, sponsoring The Combat Zone since May '06. Maybe you've been reading the build reports, manufacturing articles, and event reports since the beginning, or maybe this is your first realization that robot combat didn't die with the television series. Either way, the question has to be asked: "Are you building yet?" If not, why not?

In this article — cleverly crafted after those FAQ sections on those page things on those Internets — we'll use totally made up questions to provide real, useful answers. (Unlike virtually every "real" FAQ site ever posted, in this writer's humble opinion.)

How Do I Get Started?

Believe it or not, the answer isn't "build something!" Turns out it isn't easy to build a bot that will: a) work at all; and b) survive more than 14.1 seconds. Really, the best thing to do is go to an event, check out the pits, and talk to lots of builders. After seeing what really happens in the box, you get a whole new appreciation for the term "robust design."

While there aren't as many events as there were a few years ago, there were more than 30 held

last year in the US, Canada, Australia, and the UK. Many are listed on the Events tab at **www.buildersdb.com**. A couple organizations hold large, regular US events. Northeast Robotics Club (**http://nerc.us**) and the granddaddy of them all — RoboGames (**http://robogames.net**) — are both incredible events to attend.

So, Do I Just Show Up At An Event?

Sure. Lots of people do. Better suggestion: volunteer. No experience required, and no event EVER has enough willing hands. There is no better introduction to the builder community than doing event tasks, so they can fight and repair bots. Contact the Event Organizer and you'll be amazed at how grateful they are.

Can I Bring My Kids?

Oh yeah. Many, if not most, fighters are parent/children teams, or teenagers. A sprinkling of retired folks, rounded out by the (ahem) middle aged set, make up a diverse and good natured community. However, one thing must be made clear: These are dangerous machines. Kids MUST be under adult control at all times. The pits are often open to the public and full of sharp, heavy, high energy, hot, cold, tempting things. There is no quicker way to parental hero-dom than taking your family to a place where cool machines destroy each other.

So, I'm Hooked. Darn You, Anyway. Now What?

Build, build, build. Once you've seen the vision, build it. Most veterans recommend a first bot focus on survivability. Awesome weapons will come with experience, but a simple wedge or box bot turns out to be hard enough to build and fight. First-time machines with complex weapons usually don't do so well, break right away, and lead to frustration when they can't be repaired during an event.

Where Do I Buy The Kit?

Good question. Often asked. The answer: Well, there are a few out there. The Robot Marketplace (**www.robotmarketplace.com**) has a line of starter packages. Kitbots sells a nice, small bot platform (**www.kitbots.com**). A high-end line — BattleKits — provides a very sophisticated starting point (**www.battlekits.com**). However, most folks just sort of start on their own with a combination of components bought new, used, scrounged, and salvaged. Pretty much every builder starts clueless, gets creative, learns from experience, and quickly gets better.

So, How Do I Learn To Build My Own?

The #1 way is to buy a subscription to *SERVO Magazine* and faithfully read The Combat Zone every month. Seriously, this section is devoted to — and supported by — many of the best builders in the business. A second, invaluable resource is the *RioBotz Combots Tutorial* which is pretty much the Holy Repository Of Building Knowledge. (**www.riobotz.com**). A classic is Grant Imahara's *Kickin' Bot: An Illustrated Guide to Building Combat Robots* which is one of many titles on the subject at Amazon.

There are also a number of tutorials on various club pages. One highly recommended (although dated) is "Insect Bots for Newbies"

Photo by Sam Coniglio.

by yours truly (**www.battlebeach.com/ledgend.html**).

The best single source of advice is on the Robot Fighting League forum (**http://forums.delphiforums.com/THERFL/**) — a quirky and antiquated place full of quirky and contemporary builders with a high tolerance for newbies. Well written, thoughtful posts are quickly and helpfully answered. (Annoying and stupid posts ... well, like all forums, not so much.)

What Does It Cost?

Like all hobbies, exactly 10% more than you have to spend. Depending on whether you buy all new parts and pay for machining, or you scrounge free parts from junkyards, the costs will vary. This subject is kicked around a lot among builders. It's hard to get into the small, insect class bots for less than $300 without some creative shopping. Mid-sized bots cost easily over $1,000. Heavyweights and above, $5,000 and up.

That's not to say that many folks don't get by for less. Also, since most of the up front cost is in radio control systems and battery chargers, the bots themselves aren't the total cost. Best advice, again, is talk to builders. Most have used equipment laying around they are often eager to sell so they can buy the latest shiny object (speed controller, gearbox, machine tool) that catches their eye.

How Dangerous Is This?

Not at all. And horribly bad. It depends on how you handle it, and how the event is run. The community prides itself on a virtually spotless safety record. Arenas are designed to contain the most dangerous bots. Procedures are in place to mitigate the hazards to spectators and builders. But, like all sports, there are dangers, and it's ultimately up to each individual to follow the rules and be careful. Like woodworkers and rugby players, you'll see every builder with a band-aid on his or her hand. That is almost always the result of a shop accident — not something they got during a fight. During the repair frenzy in the pits, however, it's easy to get cut if you're not careful. So, take that into account if you have a medical issue.

So, Aren't These Folks, Like, Nasty, Mean, And Smelly To Be Around?

Actually, no. One of the real attractions of robot combat is the incredible support competitors provide each other. Stories abound of people cutting parts out of their wrecked machines to help the bot that beat them into the next round. Often, builders pitch in to repair other's bots, just to keep the competition going. Like all off-beat hobbies, a few "special and unique" folks are part of the crowd. However, without exception, combat robot fighters are there as much for the people as for the smell of magic smoke leaking out of wrecked, $150 battery packs.

Make no mistake about it, though. These folks are there to win. Once the doors on the box shut and the ref yells "Fight, Robots, Fight!" aggression is king. Some of the nastiest fighters are the youngest, with eight and nine year olds humiliating 50 year olds with their driving skill and aggressive attacks. Once the fight is over, there are traditional handshakes and congratulations both ways, with no hard feelings. This is no place for timid drivers, but there's lots of support and good sportsmanship.

Can I Organize My Own Event?

Sure. But ... we STRONGLY recommend you attend one and fight at one first. Besides learning how to run an event, this can be a very dangerous sport if proper safety precautions aren't met. Once you're ready, you'll find plenty of EOs willing to give tips, tricks, and tools to help.

Why Haven't I Done This Yet?

Don't know. Why haven't you? **SV**

All photos courtesy of RoboGames (**http://robogames.net**)

EVENTS

Upcoming Events for April 2011

Central Illinois Bot Brawl 2011 will be presented by the Central Illinois Robotics Club in Peoria, IL on April 2, 2011. For further information, please go to **http://circ.mtco.com**.

SeattleBotBattles 9 will be presented by Western Allied Robotics at the Seattle Center, in Seattle, WA on April 10, 2011. For further information, please go to **www.westernalliedrobotics.com**.

HORD Spring 2011 "Taxes and Death" will be presented by the Ohio Robot Club in Brecksville, OH on April 16, 2011. For further information, please go to **www.ohiorobotclub.com**.

RoboGames 2011 will be presented by combots in San Mateo, CA, on April 15–17, 2011. For further information, please go to **http://robogames.net/competing.php**.

Gulf Coast Robot Sports-7 will be presented by Gulf Coast Robot Sports in Bradenton, FL on April 23, 2011. For further information, please go to **www.robotmarketplace.com/products/GCRS.html**. SV

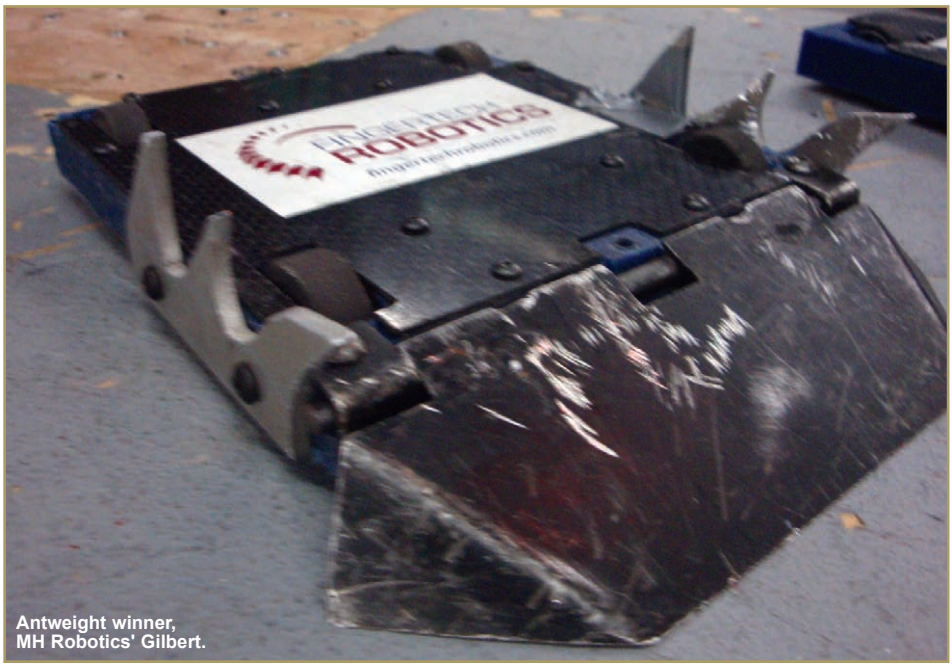
EVENT REPORT: 2011 Chattanooga Robot Battles

● by Thomas Kenney

The 2011 Chattanooga Robot Battles event was held on Saturday, January 22nd at the Chattacon Sci-Fi convention in Chattanooga, TN. This was the fourth consecutive year the event was held, and this year's competition yielded one of the biggest crops of new bots and builders yet.

The fighting kicked off at about noon with a fierce antweight double-elimination tournament. David Rohr's solid wedge, 'The Wrath of Gandhi,' proved the effectiveness of its own brand of passive aggression,

Antweight runner-up, The Wrath of Gandhi, prepares to settle the score with Rippy.



Antweight winner, MH Robotics' Gilbert.

managing to defeat one of several sacrificial RadioShack R/C cars before suffering a swift wheelectomy from the deadly saw of 'Rippy,' accompanied by a punt into the push-out box. With the new builders gradually increasing driving ability, The Wrath of Ghandi rebounded from the loser's bracket, eventually managing to defeat Rippy — along with several others, including the dreaded "Tigger," on its climb up to the final round.

Armed with its infamous horned appendages of despair, MH Robotics' "Gilbert" continued its usual success, tearing through the winner's bracket with a series of decisive push-out victories against "Segs," Tigger, and the recently christened "Green Avenger." Gilbert's teammate — a strikingly similar ant wedge and test-bin for Fingertech's new Silver Spark motors — wasn't as successful, suffering some radio and electrical issues leading to defeat at the vicious hands (saw?) of Rippy and Segs. The ant final was a continuation of Gilbert's winning streak, with a brisk five second victory as it ejected the underweight, two wheeled wedge from the arena.

The five bot beetleweight round-robot tournament followed. Team Found Object 'Bots from Atlanta, GA proved the reliability of their high speed drum weapon on the UHMW brick, "Homewrecker," but failed to rise to the top of the bracket after an unfortunate driving error in its match with the vicious, but immobilized horizontal spinner, "Misdirected Aggression," whose only true victim that day was the audience-driven toy it blew to pieces during the beetle rumble.

The Found Object group went back to their roots with their other beetle, "Blooper Reel." The joke robot's film canister shell proved more durable than most of its competitors would have hoped, and the massive 3-lber landed itself a spot in the finals against Evil Robotics' coincidentally named "Double Feature," an almost Frankensteinal mash-up of parts from Jason Brown's last couple beetle competitors bearing a small saw on the back, a drum weapon on the front, and a pair of rugged tank treads. Blooper Reel's unlikely success continued, and the tin armored bot took home the beetle crown.

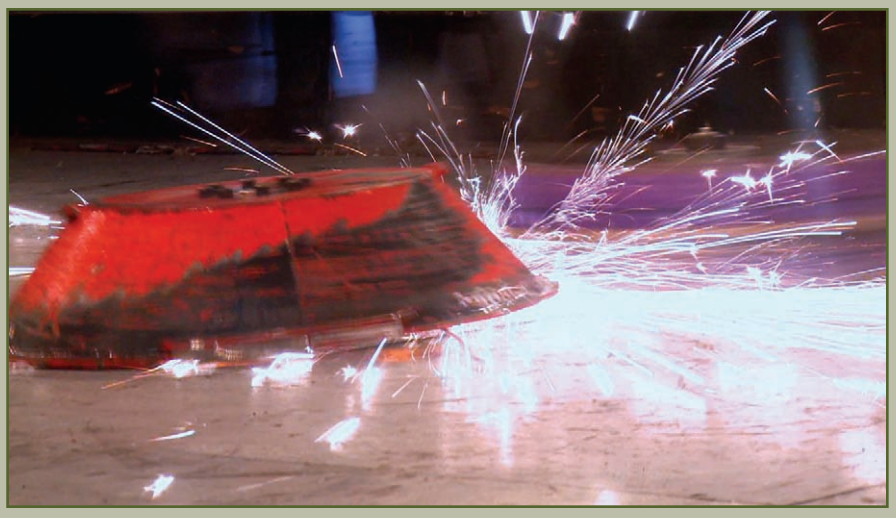
For more information on other upcoming competitions, visit **robotbattles.com**. SV


Beetleweight winner, Found Object Robots' Blooper Reel.

VIDEO REVIEW: Bots High

● by Collin Berry

Joey Daoud, a BotsIQ veteran and film school graduate, has produced a documentary about the sport. SERVO was given the opportunity to do a pre-release screening. We enlisted a journalist of similar age and background to review this interesting video.



"It's like a football game for nerds," is said early on in Bots High, a documentary about robot combat events. And that rings true throughout the entire documentary. Bots High follows two high school combat robot teams from the Miami, FL area. The two teams — a unisex team from Ransom Everglades and an all-female team from Carrollton School of the Sacred Heart — are rivals, but in a friendly way. There's plenty of trash talk between the two teams, but it's done in a friendly way — the type of trash talk you'd hear between friends at a pickup basketball game at the Y.

The documentary starts off with a bit of much expected destruction. Fluffy, a 15 pound robot from the Ransom team, is fighting a match at a regional competition. Fluffy is a very impressive robot that has a massive drum for a weapon — perhaps too massive as it proceeds to catch on fire, filling the room with smoke and causing everybody to evacuate outside.

The two teams are introduced as they prepare for the national level competition. Will is on the team from Ransom, and he is basically the student leader of the team. Will appears to be very serious when he's introduced, but it's revealed quickly that he is as carefree and fun-loving as any high school student. Sacred Heart is an all-female Catholic school. Being that they're an all-girl team, they face the type of discrimination that one might expect them to face in a male dominated hobby. However, that discrimination doesn't seem to affect them one way or the other. They're a group of girls that love building robots and are quite good at it. Famous Last Words — a 120 pound robot — has a frightening saw-type weapon.

The first half of the documentary is about the preparation for the nation tournament, and is too slowly paced. Sacred Heart builds all of their robots at a facility called Starbot. Starbot is a facility intended to get students involved in engineering. Sacred Heart has to build their robots there because their school lacks that type of facility. Ransom does have a shop, and Will has a shop in his garage, but he and the team from Ransom spend a lot of time at Starbot anyway. Will is not as motivated as previously indicated, and his robots struggle because of that.

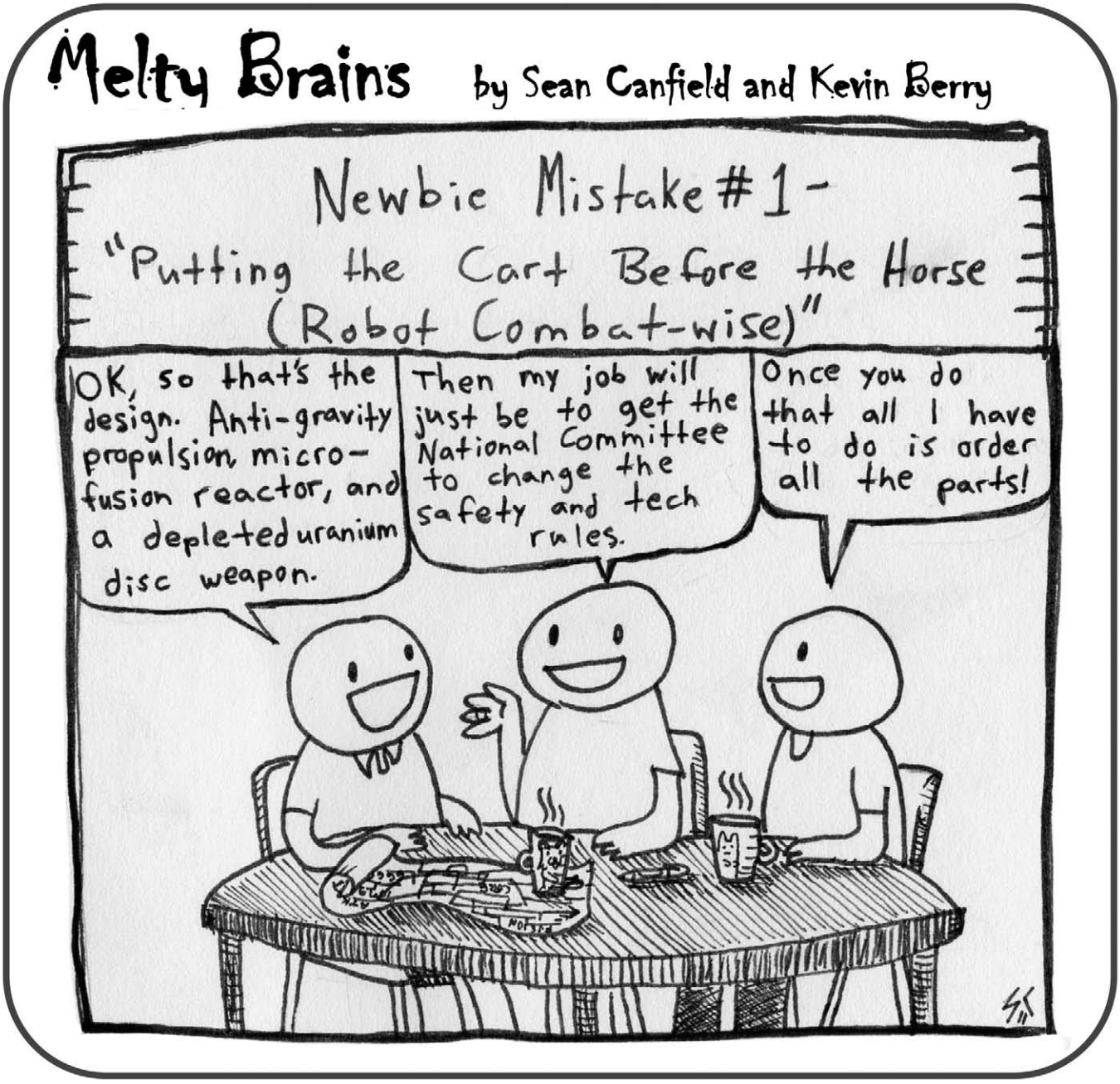
The second half of the documentary picks up as the competition arrives. Ransom shows up without a single working robot, due to procrastination and lack of motivation. Instead of relaxing, they have to build their robots. They succeed, but not without postponing their check-in and forfeiting their first fight.

Capturing the mood and spirit of the competition is where this documentary really shines, and I would have preferred more focus on it instead of preparation. There is a convivial atmosphere to the competition which is reminiscent of every robot competition I've ever attended. Teams are quick to wish each other luck, apologize for destroying somebody's robot, and help each other with quick fixes. After a weapon problem with Famous Last Words, both teams swarm it in an attempt to get it working before the next fight.

At robot competitions, there are really two competitions: one against the other robots and another against yourself. The robot has to work in the fight, and sometimes getting it to function properly is more of a challenge than the fight itself. Having a working robot, surviving all the fights, and successfully making last minute repairs can be as rewarding as winning — but that doesn't stop everybody from wanting to win. However, winning isn't the only reason the students build robots. They do it because they're skilled at it. They do it because it's a challenge. They do it because it's fun.

Bots High would appeal to anybody that is currently involved in combat robotics. It does a good job of capturing the trials that go into building a combat robot and the atmosphere of a competition. Unfortunately, I'm not sure that people that do not have a prior interest in robotics will enjoy it. It's much more of a specialized documentary.

A trailer is available at **www.botshigh.com**. SV



Making Robots With The Arduino

By Gordon McComb

Part 6 – Follow That Line!

You want simple. You want cheap. You want effective. And, of course, you want fun. That pretty much sums up line following — probably the simplest of all navigation systems for mobile robots. The basic idea goes back centuries, and is well known to anyone who's taken a train ride. Only in robotics, instead of a mechanical track the vehicle stays on course by shining a light onto a piece of tape that's stuck to the floor. Pretty advanced stuff.

No, really! Many high-tech factories use line following, where a predefined path is marked on the ground. The path can be a painted black or white line, a wire buried beneath a carpet, or any of several other methods. The path to follow can be changed simply by peeling off the tape and laying down a new stripe.

You can easily and inexpensively incorporate a tape-track navigation system in your robot. The line following feature can be your bot's only means of intelligence, or it can be just one part of a more sophisticated machine. You could, for example, use the tape to help guide your robot back to its battery charger nest. (Beats clicking your heels three times and saying, "There's no place like home ...")

So with that in mind, let's talk about line following in this installment.

ArdBot Update

Like all expandable robots, the ArdBot is designed as a permanent work in progress. It's never truly finished. In last month's article, I demonstrated a method of depowering the ArdBot's Sharp infrared proximity detector, as a way to conserve power when it's not needed. As I've refined the Ardbot's software, I have determined the method is not necessary. Turns out that in practice, it slows things down considerably and introduces as much signal noise as it eliminates.

If you've been following in the construction of the ArdBot, just skip the extra control electronics to the Sharp sensor and wire it directly. For more information on this and other enhancements for the ArdBot, see the ArdBot Construction Notes at **www.robotoid.com/servomag**.

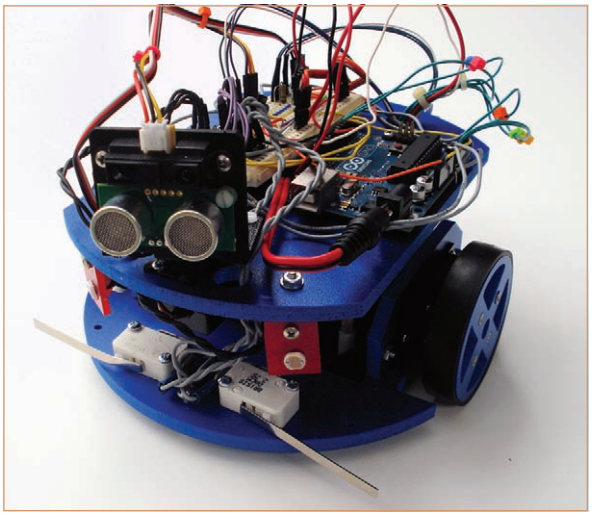


FIGURE 1. The ArdBot complete with all hardware detailed in this and previous installments.

About This Project

As with the previous parts in this series, this article builds on earlier issues of *SERVO*, starting with November '10. From there, you'll find useful information about building and programming the ArdBot which is a low-cost expandable desktop robot that uses an Arduino Uno (or compatible) microcontroller as its brain.

The ArdBot is shown in its fully "decked out" form in **Figure 1**. The pictured version incorporates all of the features we've talked about individually in previous installments, plus the line following functionality of this month's article. The ArdBot can be constructed out of plywood, plastic, even picture frame mat board. It's designed to

allow easy prototyping — it's a robot made for your own freedom of expression.

In last month's issue, I mentioned this article would conclude the series. Well, turns out there was too much to cover in one last installment. So, this time around we'll tackle just the line following component, and then conclude with putting all the pieces together. Next time, we'll combine what you've learned to create a fully-capable dual-function robot that follows predescribed paths, and explores rooms and seeks out information about its environment.

The Basics of Line Following

With a line following robot, you place white, black, or reflective tape on the floor — the color of the tape is selected to contrast with the floor. Using simple optics, the robot then follows that line.

Black PVC electrical tape absorbs infrared light, and it's cheap and easy to use. So, the most popular method of making a track is to lay down a path of electrical tape onto white paper, like that in **Figure 2**. I prefer using a stiff poster board that's coated with a smooth finish on at least one side. The coating reflects more light, and helps prevent the paper from being transparent to the infrared light. Try the nearby dollar store for inexpensive poster board. I get mine for 50 cents a sheet. Each sheet measures 22 x 28 inches. It comes in colors, but you want to start out with white.

For the best results, the floor should be hard, like wood, concrete, or linoleum, and not carpeted. You can also use a large table. To sense the line, optical sensors are placed in front of or under the robot. These sensors incorporate an infrared LED (*emitter*) and an infrared phototransistor (*detector*). The output of the phototransistor indicates whether or not it sees light reflected off the floor. Assuming a black line on a white floor, the absence of reflected light means the robot is over the line.

You can use most any infrared LED and phototransistor for the emitter and detector. Mount them close to one another, and point them in the same direction. All you need to finish the circuit is a couple of resistors, as shown in **Figure 3**.

Let's take a closer look at the components used in making a line following circuit.

Emitter LED and Current Limiting Resistor

The electronics for line following are technically two separate circuits connected to the same power source. The first part is the emitter LED and its resistor. The purpose of the resistor is to limit current flowing through the LED. You select the value of the resistor based on how much current you want to pass through the LED. The more current, the brighter the LED will glow.

Ideally, the LEDs you use come with specifications, so you know things like forward voltage drop and maximum forward current. With this info, you can use some simple math to calculate the value of the resistor you need. Here's the formula:

$$R = V_{in} - V_{drop} / I_f$$

V_{in} is five volts. Assuming V_{drop} is 1.7 volts (kinda typical for infrared LEDs), and a desired 30 mA current, the formula becomes:

$$110 = 5 - 1.7 / 0.030$$

If the result isn't a standard resistor value, always choose the next highest. In this case, 120 Ω.

(Notice I_f is given as a decimal fraction. The formula expects amps, but these LEDs are rated in milliamps. You need to move the decimal point over to convert amps to milliamps; a value of 0.030 is 30 milliamps.)

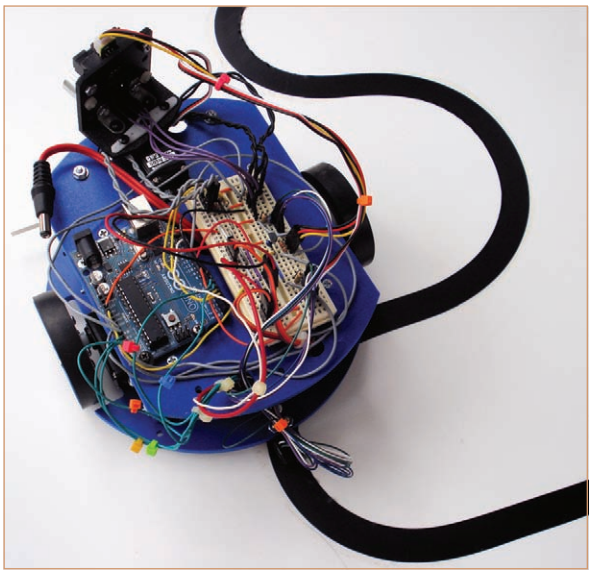


FIGURE 2. Black PVC electrical tape on white poster board makes an excellent line following course.

If you like to buy from surplus (like I do), you may not have any specs for the infrared LEDs you use. Instead, you can apply some educated guesswork to select a resistor value that's close to what you need. I usually start with a 470 Ω resistor, as this provides enough drive current to allow the LED to glow, but not so much that it will burn it out. Depending on the LED, you may need to reduce the value of the resistor, knowing that if you get too low (say,

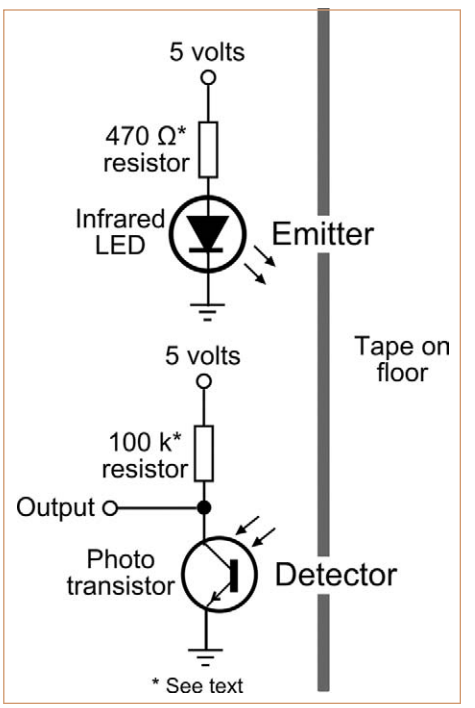


FIGURE 3. The basic line following circuitry has two parts: emitter and detector. Light from the emitter reflects off a surface and into the detector.

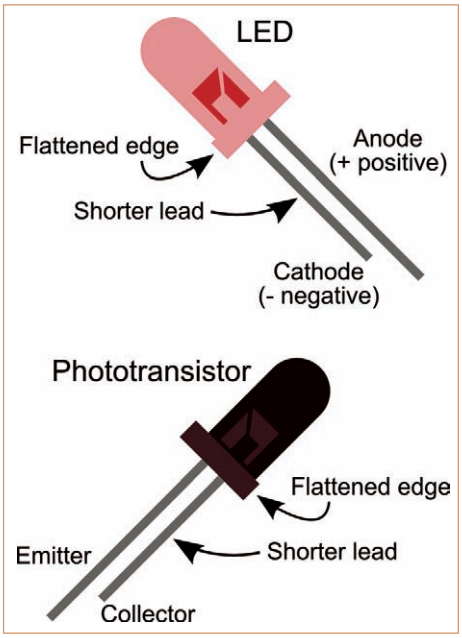


FIGURE 4. Detail of a typical LED and phototransistor T-style package. Use the short lead and/or flatted side to determine the pinout.

under 150 Ω or so), the chances of burning out the component dramatically increase.

Fortunately, many IR LEDs are engineered for fairly high maximum forward current, so you have a bit of wiggle room. While the average visible-light LED might have a maximum forward current of 30 mA, the typical IR LED is often rated higher, some 50 mA or more. (Of course, there are always exceptions. For the sake of science, you should be prepared to blow out an LED now and then.)

Should you always choose to drive the LED "hard" with lots of current? Not necessarily. For one thing, doing so ends up consuming more battery power. It can also cause unnecessary light spoilage — why use a searchlight when a candle works just as well? To start, always go with a lower current, and then decrease the value of the resistor if your circuit isn't responsive enough.

If you have a choice, pick an LED with a narrow viewing angle. It shines more light into one spot. The generic T-1 size (3 mm diameter) water-clear LED is a good choice. These have a built-in lens, and are both plentiful and cheap. They're also easy to mount by drilling a 1/8" diameter hole into the bottom of the robot.

Is That Infrared LED On or Off?

How do you know if an infrared LED is actually emitting infrared light? Your best bet is to use the camera in your mobile phone. Most of these cameras are at least partially sensitive to the infrared light wavelengths produced by IR LEDs (typically about 800 to 950 nanometers).

Activate the LED and point the camera at it. Look at the scene from the digital viewfinder of the camera. If the IR LED is working, you'll see its light as white or pale blue. **Figure A** shows the infrared emitters mounted on the underside of the ArdBot. The picture was taken using an Apple iPhone.

If the camera isn't picking up anything, do one last test to ensure it can see IR light. Point any infrared remote control at the camera and press some buttons. You should see a series of bright flashes.

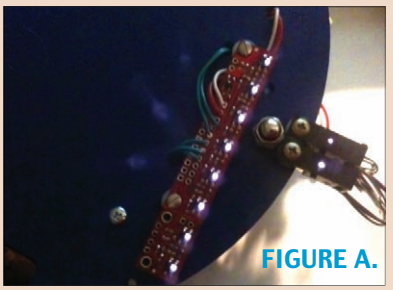


FIGURE A.

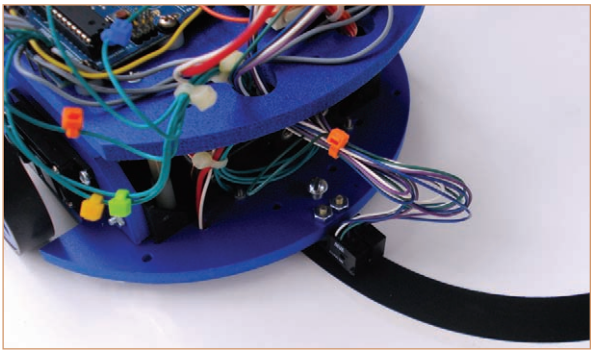


FIGURE 5. Two "store bought" reflective sensors mounted on the leading edge of the ArdBot. They're used to follow a 3/4" line made of black electrical tape.

Phototransistor and Divider Resistor

The second half of the line follower circuit is composed of a phototransistor, along with a resistor that forms a voltage divider. The value of this resistor depends on the characteristics of the phototransistor and the circuit you're connecting to; the higher the value of the resistor, the more sensitive the transistor becomes to variations in light intensity.

The easiest way to determine the value of the resistor is through empirical testing. Wire the phototransistor as shown in **Figure 3**, starting with a 10 kΩ resistor. Connect a voltmeter between the output and ground connections. Use an incandescent (not fluorescent or LED) desk lamp to alternately shine or block light to the phototransistor. You should see the voltage fluctuate on your meter. It may not be much.

Now try incrementally larger resistance values until you observe a good voltage swing — the difference in voltage between darkest and lightest conditions. It won't be the full zero to five volts of the power supply, but the larger the difference, the better. You don't need (or want) to select ever-higher resistance values when a lower one works just as well. For the phototransistors I used for my prototype ArdBot, I found good sensitivity and voltage swing with 100 kΩ resistors.

What if the phototransistor still doesn't respond well to light, even when using very high values? The phototransistor may be bad, or it may be connected in reverse. Try flipping it around in the circuit or select another one. (See **Figure 4** for how to determine the leads of the typical LED and phototransistor. Note that in a phototransistor, the flatted side/short lead is usually the collector.)

Like LEDs, phototransistors come in a variety of package styles. Look for the same package type you use for the LED. A phototransistor in a T-1 package has its own built-in lens, and it's easy to mount through a hole in the bottom of your robot.

Arranging the Infrared LED & Phototransistor

The LED emitter and phototransistor detector should be mounted side by side, and pointing in the same direction. To avoid light "crosstalk" — light that shines from the side of the LED and directly into

the phototransistor — add some black heat shrink tubing around the LED, the phototransistor, or both. Or, place a small piece of metal or opaque plastic between the two to act as a baffle.

While using separate LEDs and phototransistors is the least expensive method of creating a line following sensor, an easier approach is to use an all-in-one reflective sensor. These come in various shapes and sizes, and are fairly common on the surplus market.

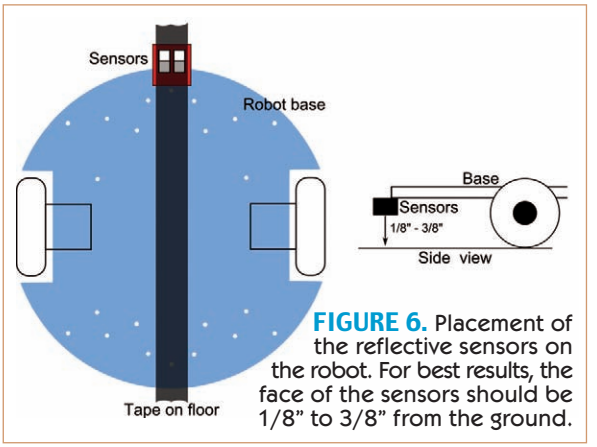


FIGURE 6. Placement of the reflective sensors on the robot. For best results, the face of the sensors should be 1/8" to 3/8" from the ground.

FIGURE 7. Parallax QTI reflective sensor, useful for line following (and other things). Note that the output of this sensor is a pulse, not an analog voltage.

Figure 5 shows a pair of reflective sensors mounted on the outside edge of the ArdBot. These sensors — which I bought surplus who knows when — provide their own mounting flange (many do; you just have to be patient and look around). They're easy to use, but because this style of reflector is bulky, they can't be mounted under a low profile robot such as the ArdBot. Instead, they have to be placed on the outside edge. That's okay, as it demonstrates one of many ways to perform line following.

Since I'm on the subject of mounting, reflective sensors like to be as close to the ground as possible, while not making actual contact. As noted in **Figure 6**, the ideal distance from the sensor to the ground is 1/8" to 3/8", depending on the design of the sensor. I mounted the two sensors to the underside of the ArdBot using plastic spacers, in order to decrease the distance from sensor surface and floor.

Several online sources offer unitized emitter/detector modules, like the QTI sensor from Parallax shown in **Figure 7**. The resistors and all components are mounted on a small circuit board. Simply attach wires to the module: power, ground, and signal. A single hole allows you to mount the module almost anywhere on the robot.

Another variation is the emitter/detector array which combines two or more pairs of infrared LEDs and phototransistors on a single circuit board. All components are already on the board, and the LED/phototransistor pairs are spaced 3/8" to 1/2" from one another — ideal for line following. In a bit, you'll see how to use one of these arrays with the ArdBot.

Whether you use homebrew line following sensors or rely on ready-made modules, the emitter and detector can be aligned either in column or row orientation, whatever fits best. By their nature, line following requires at least two pairs of emitters/detectors. These are placed a specific distance from one another, in order to determine the location of the line under the robot.

(In a pinch, you can even share one LED and several phototransistors. In this scheme, the light of the LED provides a large enough spot for two phototransistors — one on each side. This method is not recommended though, unless space is extremely limited.) The spacing of the emitter/detector pairs depends on the width of the tracking tape. To be effective, at least one sensor should detect the line at any one time. With too much space, a thin line can get "lost" between sensors. You can fix this either by grouping the emitter/detector pairs closer together, or using a wider tape. **Figure 8** shows some variations. As most, PVC

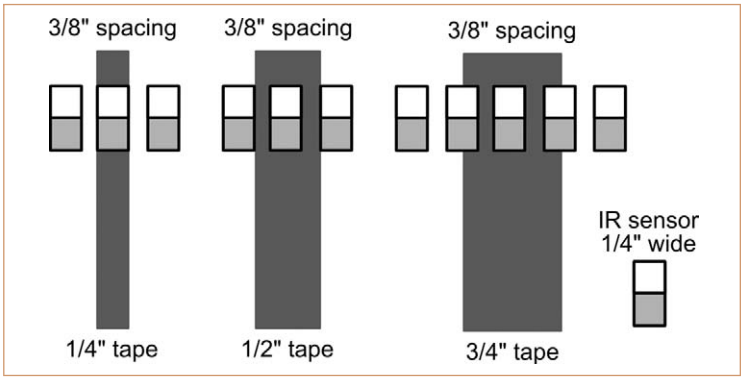


FIGURE 8. The spacing between reflective sensors is dependent on the width of the line. The line must be visible to at least one sensor at any time.

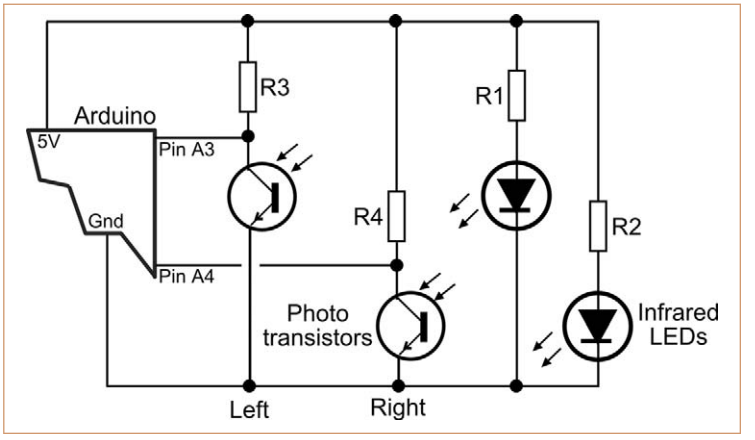


FIGURE 9. Circuit for connecting two LEDs and phototransistors to the Arduino. See the text for the values of the four resistors.

electrical tape is 3/4" wide; a spacing of 3/8" to 1/2" is most common.

By the way, when we talk of spacing, it means the distance between the optical centers of each emitter/detector pair, regardless of the physical dimensions of the sensor. Given sensors that are physically 1/4" wide and 3/8" between optical centers, the actual spacing between each component is only 1/8".

Listing 1

```
/*
 ArdBot line following demo using
   2 reflective sensors
 Requires Arduino IDE version 0017
   or later (0019 or later preferred)
*/

#include <Servo.h>

Servo servoLeft;              // Define left servo
Servo servoRight;             // Define right servo
const int lineLSense = A3;
const int lineRSense = A4;

int irReflectR = 0;
int irReflectL = 0;
int thresh = 400;

void setup() {
   servoLeft.attach(10);   // Left servo pin D10
   servoRight.attach(9);   // Right server pin D9
}

void loop() {

   // Read reflective sensors
   irReflectL = analogRead(lineLSense);
   irReflectR = analogRead(lineRSense);

   if (irReflectL >= thresh && irReflectR >= thresh) {
      line_forward();    // on line
   }

   if (irReflectL >= thresh && irReflectR <= thresh) {
      line_spinLeft();   // veering off right
      delay(4);
   }

   if (irReflectL <= thresh && irReflectR >= thresh) {
      line_spinRight();  // veering off left
      delay(4);
   }

   // If line is lost try to reacquire
   if (irReflectL < thresh && irReflectR < thresh) {
      line_spinRight();
      delay(20);
   }

}

// Motion routines for line following
void line_forward() {
   servoLeft.write(0);
   servoRight.write(180);
}

void line_slipRight() {
   servoLeft.write(90);
   servoRight.write(180);
}
void line_slipLeft() {
   servoLeft.write(0);
   servoRight.write(90);
}
void line_spinRight() {
   servoLeft.write(180);
   servoRight.write(180);
}
void line_spinLeft() {
   servoLeft.write(0);
   servoRight.write(0);
}
```

Constructing a Two-Sensor Line Follower

The most basic line follower uses two emitter/detector pairs. To work properly, the pairs must be close enough together to both see the line when the robot is directly over it. If the left sensor doesn't detect the line, it means the robot has veered too far to the right. To compensate, the robot steers back over to the left. Conversely, if the right sensor doesn't detect the line, the robot corrects its path by briefly steering back toward the right.

Figure 9 shows the schematic view of connecting two IR LEDs and two phototransistors to the Arduino. Since the output of the phototransistors is a variable voltage, we use two of the Arduino's analog-to-digital converter inputs; specifically, pins A3 and A4. **Figure 10** shows the same circuit but in breadboard view. (The bottom of the breadboard is dedicated to the servo wiring. See Part 2 of this series for more information.)

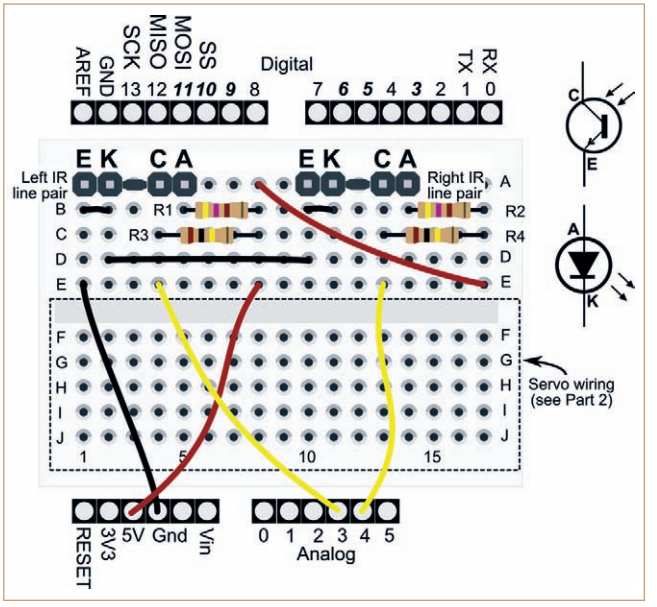
Resistor values to start (see the previous discussion for selecting the values based on the LEDs and phototransistors you use) are:

R1, R2: 470 Ω
R3, R4: 100 kΩ

Listing 1 shows a simple line following sketch. After setting up the two servos, the analog values of the two sensors are read. These values are then applied to one of several actions:

- Line is detected by both sensors; keeps going forward.
- Line is detected only by the left sensor; steer briefly to the left.
- Line is detected only by the right sensor; steer briefly to the right.
- Line is not detected by either sensor; steer in a circle,

FIGURE 10. Breadboard view of the circuit in Figure 9.



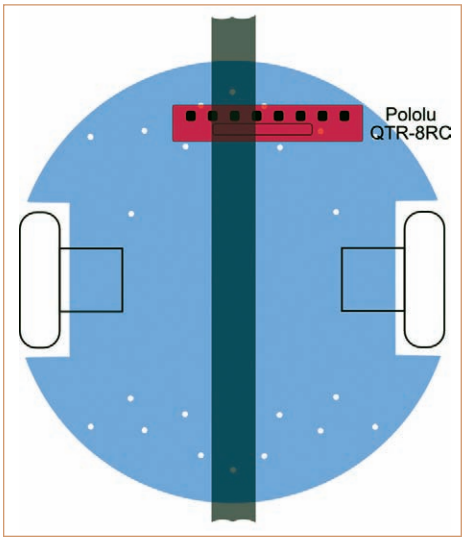


FIGURE 12. Orientation of the QTR-8RC on the underside of the ArdBot. I've offset the board so that sensor 3 is centerline. This is purely subjective, and you can mount the board in any way you choose. For a more elaborate system, use all eight sensors and mount the centerline between sensors 4 and 5.

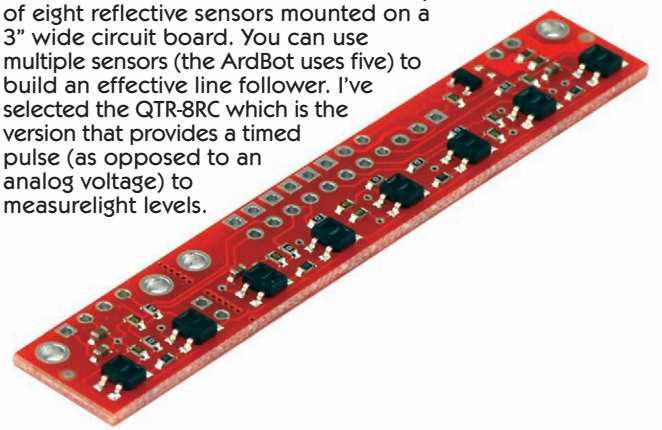


FIGURE 11. The Pololu QTR-8 is an array of eight reflective sensors mounted on a 3" wide circuit board. You can use multiple sensors (the ArdBot uses five) to build an effective line follower. I've selected the QTR-8RC which is the version that provides a timed pulse (as opposed to an analog voltage) to measurelight levels.

and attempt to re-acquire the line.

(If you use a self-contained emitter/detector module, then connection is much easier. Just attach the power pins of the module to 5V and Gnd of the Arduino, and the signal pin to either analog pin A3 or A4. If you use a self-contained module, be sure its output is analog, or else it won't work with the program in **Listing 1**. As we'll see, some variations in modules — including the aforementioned Parallax QTI — use a timed pulse rather than a varying voltage.)

Important! On a bidirectional robot like the ArdBot, front and back — and therefore right and left — are arbitrary. The sketch in **Listing 1** assumes the line following optics are actually in the rear of the robot. It's done this way because other navigation sensors on the ArdBot are mounted on the opposite end. We'll use this feature in the next installment, where the ArdBot serves as a dual-purpose bot and travels in the direction dictated by its current mode.

If your robot goes the wrong way, you can simply swap the servo connections on the breadboard; left becomes right, and right becomes left.

As I said before, I mounted the two sensors on the outside edge of the robot rather than underneath. Though there are advantages to putting the emitters/detectors under the bot, this method works too. I especially like this approach as a demonstrator because it's much easier to visually see how the robot is tracking the line.

To play with the ArdBot as a line follower, you'll need to construct a track for the robot to follow. See the **sidebar** "Building a Line Following Course" for some basic ideas.

Note the thresh variable and the numeric value I've given it. In my tests, I found that the sensor readings went from 30 when over white and 850 when over black. I picked a threshold value of 400; that's about halfway between. You'll need to calibrate the thresh value based on the sensors you use. **Listing 2** provides a simple routine for displaying the numeric values returned by an analog line following sensor.

To use it, compile and download the sketch in the usual manner, then open the Serial Monitor window. Place the sensor connected to analog pin A3 over the black line. Note the value.

Building a Line Following Course

Forget protractors, compasses, and straight edges for now. For a basic line following course — good for practice and experimentation — just pull out a couple of sheets of poster board and lay down some tape.

Using just a single sheet of poster board, begin by making a U-shape. Keep the line away from the very edges of the board. As you apply the tape give it a bit of stretch, but not too much. The tape might contract as it springs back into its former size. You want to avoid excessively tight corners. If the edges of the tape pucker up in a curve, it's probably too tight.

On the same or another sheet of poster board, make a few sinusoidal curves like the ones in **Figure B** where the radius of the curves is the radius of the ArdBot's wheel base or larger. When using spin turns (rather than slip turns) and with a sensor array like the QTR-8, you can expect the ArdBot to manage curves that are at least as small as its own wheel base diameter.

If you need a larger course, place two or more sheets of poster board end to end. Cut the tape off right at the edges of the paper. You don't need to overlap the tape for a perfectly seamless transition, but avoid gaps of more than 1/84".

Wanting to use something other than electrical tape? Feel free to experiment, but know that what looks black to you might be optically transparent to infrared light. Black marker is a good example. Unless you heavily apply marker ink to both sides of the paper, odds are it'll prove to be a poor choice for making line following tracks.

If you have a laser printer, you can experiment with making segments of tracks using a graphics program. Print out each sheet and tape them together. Be sure to use a fresh toner cartridge, and adjust the toner density to make it darker. The same technique may (or may not) work for ink jet printers. While pitch black to our human eyes, the ink is not opaque to infrared light.

Alternatives to black PVC electrical tape include black masking tape (try more than one layer thick), charting and graphics tape, and decal making tapes. While it's most common to use black tape on a white background, you can reverse the tones — just remember to also flip the programming logic in your sketch. Try using black construction paper or poster board and silver reflective tape.

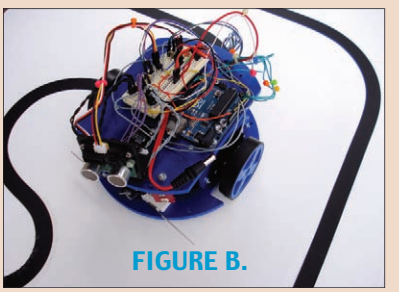


FIGURE B.

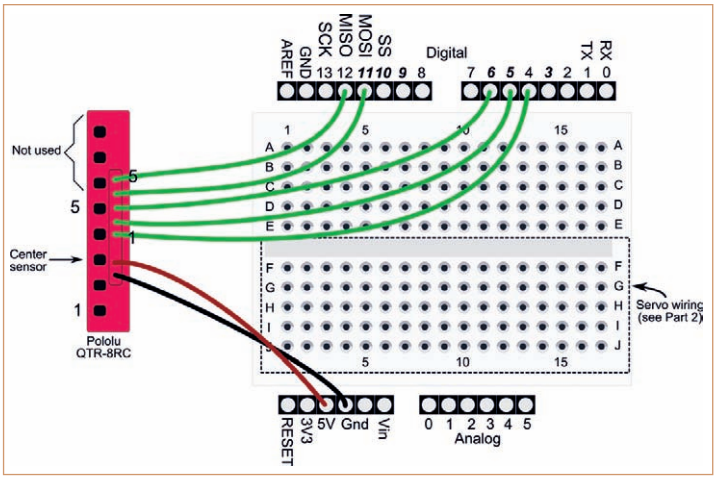


FIGURE 13. Breadboard view of hooking up the QTR-8RC to the ArdBot.

Listing 2

```
// ArdBot reflective sensor demo

int irReflect = 0;

void setup() {
  Serial.begin(9600);
}

void loop() {
  // Connect sensor to analog pin A3
  irReflect = analogRead(A3);
  Serial.println(irReflect, DEC);
  delay (200);
}
```

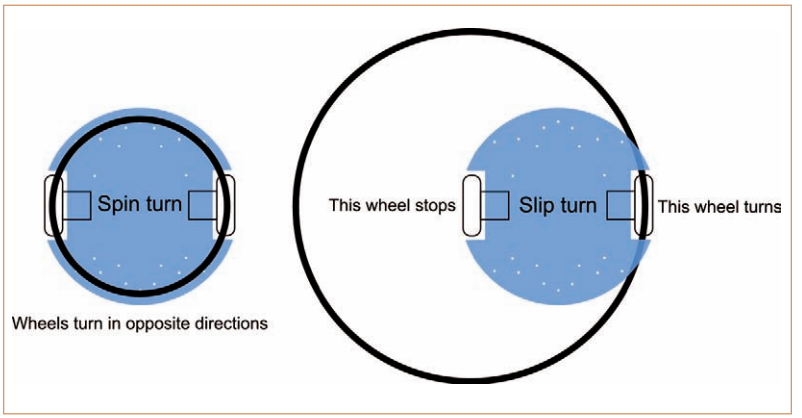


FIGURE 14. "Turning circles" when using spin turns as opposed to slip turns. In a slip turn, the circle is twice as large.

Now place it over the white background. For best results, there should be a variation of at least 200 points. (Example: If the value over black is 600, then the value over white should be under 400.) Set the *thresh* variable in your sketch somewhere between the values you obtained between black and white.

Feel free to experiment with the sketch to see how to improve the performance of your line following ArdBot. I've intentionally kept it simple to make it easier to understand. Try using the line_slip functions rather than the line_spin functions. How does this affect maneuverability of the robot?

Constructing a Five-Sensor Line Follower

The two-sensor line follower provides good results even on courses with reasonably tight turns. It offers a good starting place to try things out. With more sensors, you can better determine where the line is, was, and/or should be. Adding more sensors is a simple matter of constructing more emitter/detector pairs. If you go that route, I advise you build your own self-contained module using a solderboard to permanently connect all the components. This dramatically decreases the number of wires you must attach to the Arduino and solderless breadboard.

A better approach — and one that doesn't cost that much — is to use a ready-made emitter/detector array, like the one in **Figure 11**. This array — the QTR-8 from Pololu — contains eight sensor pairs, though for the following project we'll only use five of them. Everything comes on the surface-mount board which is thin enough to mount underneath the ArdBot (see **Figure 12**). Just attach power, ground, and signal wires to the sensor pairs you wish to use.

Note that I've lined up the sensors so that sensor 3 (third from the left) is centerline in the base. I'm only using the first five sensors, leaving sensors 6-8 unconnected (see **Figure 13**). With this arrangement, I obtain values from sensors 1 through 5, with sensor 3 directly in the middle of the ArdBot.

The QTR-8 is available in two versions: analog and RC. I've selected the RC version (part QTR-8RC), as it can be connected to either the analog or digital pins of the Arduino. The RC comes from the resistor and capacitor used with each sensor. In operation, the output of each sensor of the array provides a pulse whose length depends on the relative darkness or brightness of the light detected by the phototransistor (the Parallax QTI works in the same way). By measuring the length of this pulse, it's possible to obtain very accurate readings from each sensor. As it turns out, Pololu makes available a handy library for using the QTR-8 array with the Arduino. The library simplifies the use of the sensor array, and it's ideally suited for line following. See **Sources** for URLs for the documentation for the QTR-8 and the Arduino library (there's also a library for generic AVR microcontrollers; be sure to get the one fine-tuned for the Arduino).

The Arduino QTR-8 library comes in a zip archive file. Download it to your computer, then:

1. Exit the Arduino IDE if it's already running.
2. Find your Arduino sketchbook folder (normally it's in My Documents).
3. Look for and open the libraries folder. If there isn't a folder named libraries, create one. Extract the contents of the Arduino QTR-8 library into this folder. This creates a new folder named PololuQTRSensors.
4. Depending on the way your zip extraction program works, the files in PololuQTRSensors may be located within yet another folder.

Move the files out of this inner folder, so they are directly under PololuQTRSensors.

5. Start the Arduino IDE. In the IDE, choose Sketch->Import Library-> PololuQTRSensors. (If the PololuQTRSensors item is not shown, exit the IDE and double-check that you have unpacked the zip file to the proper place. Restart the IDE and look again.)

When you select the PololuQTRSensors library item, the following should be inserted into your sketch:

```
#include <PololuQTRSensors.h>
```

This establishes that you wish to use the PololuQTRSensors library and all its features in your sketch. **Listing 3** shows a complete demonstration of using the QTR-8RC with the Arduino and ArdBot. It uses the PololuQTRSensors library to do most of the heavy lifting. (As with **Listing 1**, this sketch assumes the QTR-8 has been mounted on what I've defined as the rear of the ArdBot. If your robot goes in the opposite direction, simply swap the left servo connection for the right.) Points of interest in this sketch include:

- It creates a single object that represents the QTR-8 sensor array. The name of the object is qtrrc. Its constructor (the programming statement that creates the object) requires a number of parameters, including the Arduino pins used for connecting to the array and the total number of sensors. I'm using digital pins 4, 5, 6, 11, and 12.
- The PololuQTRSensors library includes a method for calibrating the sensors. This method is called with the line qtrrc.calibrate(). This calibration takes place in the setup() function when the Arduino first runs the sketch. During calibration, the built-in LED on pin D13 of the Arduino lights up. This is your cue to slowly move the sensor array over the lightest and darkest areas of your line following course. This stores the minimum and maximum detected levels, and automatically adjusts for differences in the individual sensors. The LED goes out when calibration is done.
- I'm using only the position value provided by the PololuQTRSensors library which combines the results from all the sensors into one (you can get the results from the individual sensors, as well). The

Listing 3 – QTR-8RC Line Following Sketch.

```
/*
 ArdBot line following demo using
  Pololu QTR-8RC sensor array
 Requires Arduino IDE version 0017
  or later (0019 or later preferred)
 Requires PololuQTRSensors library
  (available from Pololu; see text)
*/

#include <PololuQTRSensors.h>
#include <Servo.h>

Servo servoLeft;           // Define left servo
Servo servoRight;          // Define right servo

#define NUM_SENSORS    5      // Number of sensors
#define TIMEOUT        2500   // Maximum timeout per sensor

// QTR-8 connected to digital pins 4, 5, 6, 11, and 12
PololuQTRSensorsRC qtrrc((unsigned char[]) {4, 5, 6, 11, 12},
  NUM_SENSORS, TIMEOUT, QTR_NO_EMITTER_PIN);
unsigned int sensorValues[NUM_SENSORS];
int variance = 300;         // Variance from center (2000)

void setup() {
  servoLeft.attach(10);  // Left servo pin D10
  servoRight.attach(9);  // Right server pin D9

  delay(500);
  pinMode(13, OUTPUT);
  digitalWrite(13, HIGH);       // Show calibrate mode
  for (int i = 0; i < 400; i++) {
    qtrrc.calibrate();          // Read sensors to calibrate
  }
  digitalWrite(13, LOW);        // Calibrate completed
  delay(1000);
}

void loop() {
  unsigned int position = qtrrc.readLine(sensorValues);

  if (position > (2000 - variance) && position < (2000 + variance)) {
    line_forward();
  }
  if (position > 2000 + variance) {
    line_spinLeft();
    delay(2);
  }
  if (position < 2000 - variance) {
    line_spinRight();
    delay(2);
  }
}

// Motion routines for line following
void line_forward() {
  servoLeft.write(0);
  servoRight.write(180);
}

void line_slipRight() {
  servoLeft.write(90);
  servoRight.write(180);
}
void line_slipLeft() {
  servoLeft.write(0);
  servoRight.write(90);
}
void line_spinRight() {
  servoLeft.write(180);
  servoRight.write(180);
}
void line_spinLeft() {
  servoLeft.write(0);
  servoRight.write(0);
}
```

Sources

If you'd like to build the ArdBot, be sure to start with the November '10 issue of *SERVO Magazine* for Part 1 of this series. Also check out the following sources for parts:

Prefabricated ArdBot body pieces with all construction hardware:

Budget Robotics
www.budgetrobotics.com

Arduino Resources:

Arduino
www.arduino.cc

Fritzing
www.fritzing.org

Online Retailers of Arduino, infrared sensors for line following, and components:

AdaFruit Industries
www.adafruit.com

Jameco
www.jameco.com

Lynxmotion
www.lynxmotion.com

Pololu Robotics & Electronics
www.pololu.com

RobotShop
www.robotshop.com

SparkFun
www.sparkfun.com

Pololu QTR-8 Datasheets and Arduino Library:

User Guide
www.pololu.com/docs/pdf/0J12/QTR-8x.pdf

Library Documentation
www.pololu.com/docs/pdf/0J19/QTR_arduino_library.pdf

Library files (in zip format)
www.pololu.com/file/download/PololuQTRSensorsForArduino.zip?file_id=0J403

Value	Meaning
0	Line is under sensor 1 or has moved beyond sensor 1.
1000	Line is under sensor 2.
2000	Line is under sensor 3 (the middle).
3000	Line is under sensor 4.
4000	Line is under sensor 5 or has moved beyond sensor 5.

position value indicates the relative position of the array over the line. With five sensors, the value will go from 0 to 4000.

Incremental values between 0 and 4000 are possible. These indicate the line is between sensors. For example, a value of 1500 means the line is midway between sensors 2 and 3.

Recall that I've lined up sensor 3 with the centerline of the robot. A value of 2000 means the line is directly centered under the robot. Any value less than 2000 means the robot has gone off course to the left, so it should be steered over to the right. Conversely, any value more than 2000 means it's gone off course to the right.

Even when the robot is over the line, the position value will rarely be exactly 2000. So, I've added a variance to the centerline measurement to prevent the robot from constantly "hunting" right and left. With a variance of 300, "centered" means anywhere between 1700 to 2300. You are free to play around with this value. Just change the variance variable accordingly.

Slip Turns Versus Spin Turns

Take another look at the sketches in **Listings 1** and **3**. You'll see two sets of functions for turning right and left. One set is referred to as slip, and the other as spin. Here's the difference: In a slip turn, one motor stops while the other continues. In a spin turn, one motor reverses direction while the other continues. As shown in **Figure 14**, the two approaches affect how sharp of a turn the robot can make.

In a slip turn, a differentially-steered vehicle like the ArdBot turns in a circle equal to double its wheel base. Since the ArdBot has a seven inch wheel base, that means the robot can trace the path of a 14 inch circle. If you cut out a section of that 14 inch circle to make a curve (and depending on the number and/or placement of sensors), that curve represents the tightest corner the robot can reliably manage.

As it happens, curves from a 14 inch circle are still pretty tight. However, the ArdBot can manage even better if its wheels rotate in opposite directions in a turn. In a spin turn, the vehicle can trace a much sharper path; the minimum curvature will greatly depend on the number and placement of the sensors.

Each type of turn has its place. Slip turns tend to be smoother. Only the tightest corners require spin turns, and ordinary cornering can use slip turns. It's not practical to determine if a corner is too steep when using a two-sensor line follower, but it's quite possible in systems that use five or more sensors. The further away the line is from the center sensor, the sharper the turn.

In a more detailed sketch, you could use the position information provided by the QTR-8 (or equivalent) sensor to determine which type of turn is required. For example, if the value is between 2300 and 3000, the robot need only correct its course using a slip turn. If the value is above 3000, a spin turn will do the trick.

Line Following Enhancements

There are plenty of ways to enhance the functionality of the ArdBot's line following feature, especially when using a sensor array.

For example, by placing a three to five inch strip of tape perpendicular to the end of the track, you can have your robot know when it's reached its destination. Add a routine where if four or more sensors see black, the course is finished and the robot should stop.

We'll talk about this enhancement and a few others in our next installment. In the meantime, feel free to experiment on your own. Buy an extra roll of electrical tape, a stack of poster board, and have fun!

Coming Up ...

Now with all the hardware in place, we'll finish up by putting what you've discovered to good use. You'll learn more tricks of the trade for line following, plus unearth ideas and programming code to allow your ArdBot to explore a room, looking for people, pets, and other objects to investigate. SV

VEX Stepper Motor Control Experiments

SunBot III

By Daniel Ramirez

It is said of Archimedes of Syracuse (c. 287 BC – c. 212 BC) — an extraordinary Greek mathematician, physicist, engineer, and astronomer — that he is best known for his mathematical works relating to computing the area under a parabola and also computing the value of pi accurately. Among his many discoveries and inventions were weapons which were used to successfully defend his Sicilian homeland from the Romans. According to historians, Archimedes devised island defenses to repel many Roman attacks and sieges. Legend has it that one of his defenses involved using arrays of parabolic shaped mirrors placed at locations along the shore line that were focused on Roman ships to set them on fire. Kind of like an ancient "heat ray." While not necessarily the best use of solar power, this is the basic theory behind modern mirror reflector solar power stations (heliostats) that are capable of generating electric power by focusing the mirrors on a steel tower used to super heat water into steam to generate electricity. It is amazing how much of Archimedes' work influences science, modern architecture, and technology, especially in the area of mathematics.

SunBot III

SunBot II's mechanical and electrical subsystems and stepper motor drives were described in great detail in the last article that ran in the December '10 issue of *SERVO*, leaving the firmware, quadrature encoders, limit switches, and stepper motor calibration details for this installment. I also mentioned deficiencies that I encountered with SunBot II, including backlash from the gears causing pointing errors especially in the azimuth axis. The solution to these problems was to go back to the drawing board and design and build a new SunBot III shown in **Figure 1**. In this **photo**, you can see the large stepper motors that we selected. They are powerful enough to drive the azimuth and elevation axes directly, and are mounted to the frame made from VEX components, some scrap metal plates, and a lazy Susan. The lazy Susan supports the weight of the elevation subassembly while providing very smooth motion. In addition, we used two new SparkFun v 4.3 Easystepper boards which allow these large stepper motors to be micro-stepped.

The steel rail shown in **Figure 1** serves two purposes. First, it is a mount for a laser pointer used to indicate how many degrees azimuth the elevation axis has moved. The laser pointer generates a spot below on the setting circle that indicates this. Notice that it is placed perpendicular to the solar panel, so that it would not get in the way and

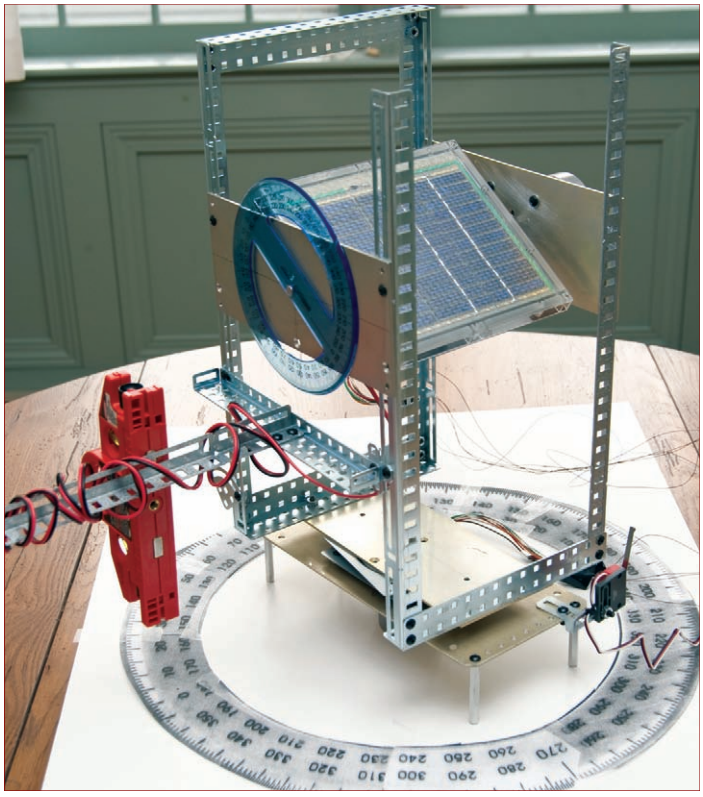


FIGURE 1. Note the larger stepper motors that are used to drive the azimuth and elevation axes directly. These are are mounted to the frame which is made from VEX components, some scrap metal plates, and a low cost lazy Susan. You can also see the altitude and azimuth setting circles used for calibration.

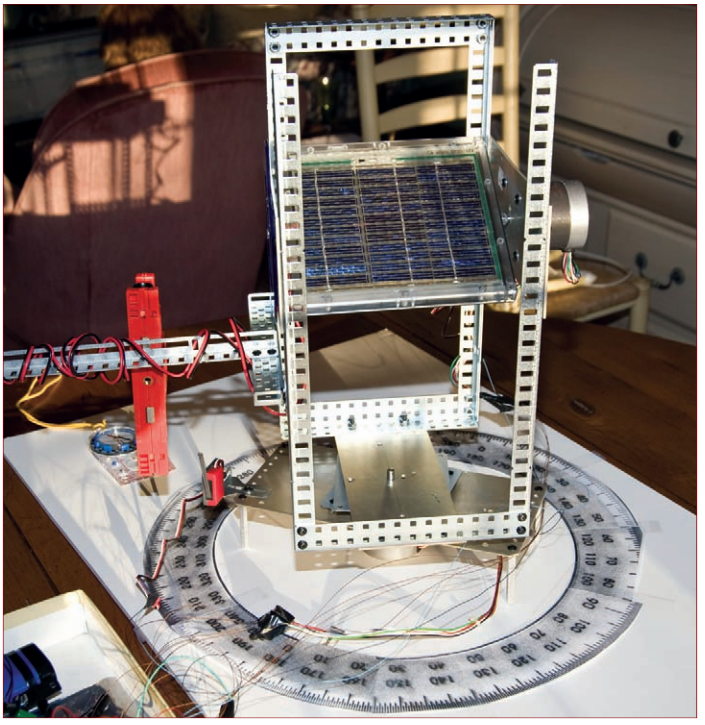


FIGURE 2. Another view of Sunbot III facing the sun. You can see all the wires leading from the stepper motors and VEX limit switches back to the microcontroller and Easystepper boards. These wires are long enough to wrap around to any position without getting snagged. You also get a clear view of the azimuth setting circle with zero to 360 degree markings.

block sunlight. This means that azimuth readings need to be offset by 90 degrees. The second purpose is to act as a counterbalance for the heavy elevation stepper motor located on the opposite side of the elevation frame. The blue circular protractor attached to the elevation axis is used to measure the elevation (altitude) angles using the protractor's 0 to 180 degree markings referenced from the horizontal line as shown in the **figure**.

The azimuth setting circle shown in **Figure 2** is similar to those used on telescopes and was constructed using a copier to enlarge a standard circular protractor (found at an office supply store) by 550%. It is used to calibrate and align SunBot and to show the azimuth. In addition, I used the setting circle on the azimuth axis to calibrate it. The compass shown in the **figure** is used to find true North which is aligned with 0 degrees azimuth. Remember to adjust for the declination from magnetic North which the compass points to.

The lazy Susan used for the azimuth axis needs to be carefully centered with the elevation axis. Insure that it does not bind when it is attached to the elevation axis via the azimuth stepper motor axle. If it does bind, then realign it and — if necessary — do not fasten it with screws to the elevation axis assembly. Just mount the assembly onto the top of the lazy Susan in order to support the weight of the elevation axis. The red limit switches shown in the **figure** provide detection of the "home" position from which to start moving the solar panel.

The updated parts list for building SunBot III is shown in **Table 1**. Notice that some of the components are optional, but were used when I developed and tested SunBot III. In fact, we only used the optional VEX quadrature encoders in SunBot II (December '10) to verify the geared stepper motor moves and as a backup for the limit switches since they did not provide enough resolution at 90 Counts Per Revolution (CPR) to be used for controlling the stepper motors. For this functionality, we really need high resolution (400 CPR to 1,600 CPR) quadrature encoders that are not currently sold by IFI.

Once calibrated, the SunBot or other mechanisms can be commanded to any position in open loop fashion or they can continue using feedback from high resolution encoders as verification that the move was correct. This is certainly true when used with surgical instruments in medical applications.

The schematic shown in

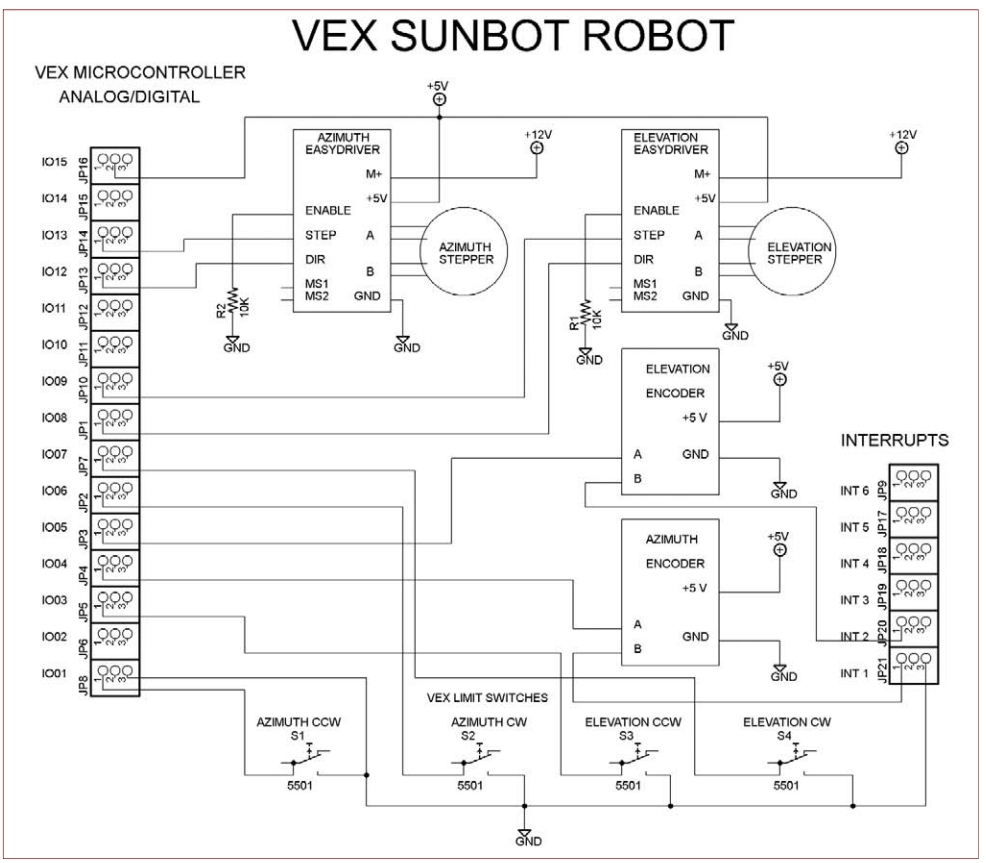


FIGURE 3. The updated schematic used to connect the SparkFun Easystepper v 4.3 boards to the more powerful stepper motors. The boards have a M+ power input to connect an external power supply (six or 12 volt SLA battery). It also shows how to connect the VEX limit switches and optional VEX encoders.

Figure 3 provides the necessary details for connecting the two SparkFun Easystepper boards, two encoders (optional) attached to each axis of motion, and the VEX limit switches used to detect mechanical hard stops. This schematic is more detailed than the one previously presented. To obtain encoder counts as feedback from each stepper motor, we can install a VEX quadrature encoder to each stepper motor axle so that we can correlate them to stepper counts during calibration, then determine if they skipped any steps.

The VEX microcontroller shown in **Figure 4** sends stepper commands to the Easystepper boards. The two red Easystepper boards are shown connected to the VEX microcontroller with long .100 pin headers with wire-wrap and jumper wires. Wires should be flexible and light and long enough to not get snagged around the metal parts, but also sufficient to carry the current to the two stepper motors.

Control of Large Stepper Motors

WARNING: The VEX microcontroller is capable of controlling very large motors which requires caution when operating since the voltages, currents, and torques are far greater than those used on standard VEX motors. It is also important to use wires that can handle the large currents drawn by the stepper motor coils and make sure that the connections are adequately insulated with electrical tape in order to avoid short circuits and fires. Also, be sure to wear goggles or safety eyeglasses and keep a safe distance from the SunBot or other mechanical structures using stepper motors.

In order to drive the two large stepper motors used on SunBot III (shown in **Figure 5**), the large azimuth stepper motor directly attaches to the base of the lazy Susan and the elevation subassembly. **Figure 6** shows the elevation drive stepper motor connected directly to the 12 volt solar panel using a 1/4 inch stainless steel rod. This stepper motor is relatively heavy so it needs to be counterbalanced.

A VEX microcontroller is capable of moving these steppers with its own battery; it is not recommended here, however, since it may not provide enough current to the Easystepper boards to move the steppers accurately. Plus it will drain the VEX battery very quickly. For this reason, it is necessary to supply an external six volt or 12 volt SLA battery and connect it to the M+ and GND power inputs of the Easystepper boards that were shown in **Figure 2**. The battery may be connected to the solar panel during recharge cycles using the output power cable leading from the solar panel. When wiring the stepper motors, make sure that you disconnect power from every board and turn off the microcontroller. Check the temperature of the wires and of the two Easystepper boards to make sure they are not too hot to the touch. If they do feel hot to the touch, then the stepper motors may be drawing too much current.

TABLE 1. Bill of Materials. The optional VEX quadrature optical encoders are used to verify the stepper motor commands to azimuth and elevation positions.

ITEM	QTY	DESCRIPTION	SOURCE
1	1	VEX microcontroller	Innovation First, Inc. **www.vexforum.com**
2	1	12 volt SLA battery	RadioShack **www.radioshack.com**
3	2	Easystepper boards v 4.3	SparkFun **www.sparkfun.com**
4	1	Wire-wrap cable	RadioShack **www.radioshack.com**
5	1	*Package of jumper cables	SparkFun **www.sparkfun.com**
6	2 or 4	VEX limit switches	Innovation First, Inc. **www.vexforum.com**
7	2	*VEX optical encoders	Innovation First, Inc. **www.vexforum.com**
8	1	VEX structural components	Innovation First, Inc. **www.vexforum.com**
9	1	VEX 9.6 volt battery	Innovation First, Inc. **www.vexforum.com**
10	1	*VEX RC transmitter	Innovation First, Inc. **www.vexforum.com**
11	1	*VEX receiver	Innovation First, Inc. **www.vexforum.com**
12	2	Large stepper motors	All Electronics **www.allelectronics.com**
13	1	Heavy duty 12 volt marine solar panel	All Electronics **www.allelectronics.com**
14	30	.100 pin headers (1" length)	Digi-Key **www.digikey.com**
15	1	Circular protractor (0 to 360 degrees)	Staples **www.staples.com**
16	1	*Laser pointer	Staples **www.staples.com**
17	1	Magnetic compass	Sporting goods store
*Items are optional.			

FIGURE 4. Two red Easystepper boards connected to the VEX microcontroller with .100 pin headers. Following the schematic, use wire-wrap and jumper cables to make the connections.

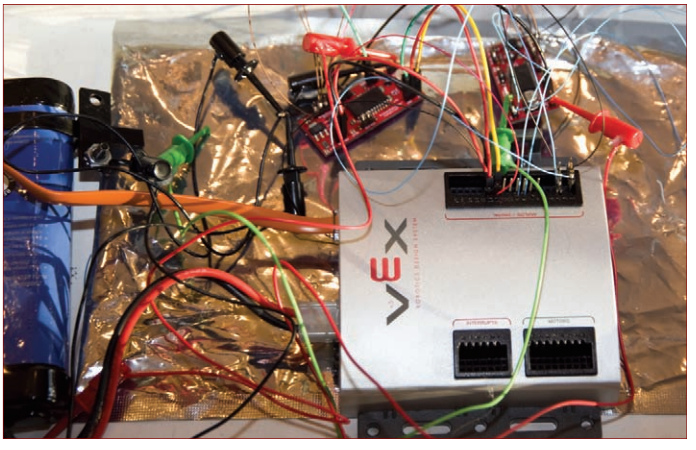
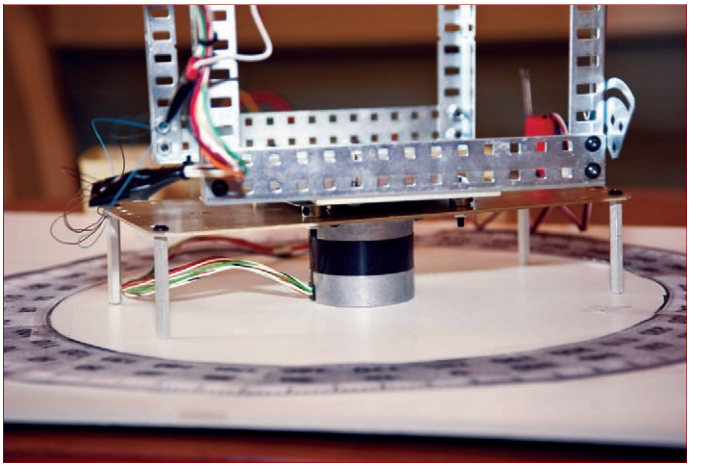


FIGURE 5. The large azimuth stepper motor directly attached to the base of the lazy Susan, and the elevation subassembly.



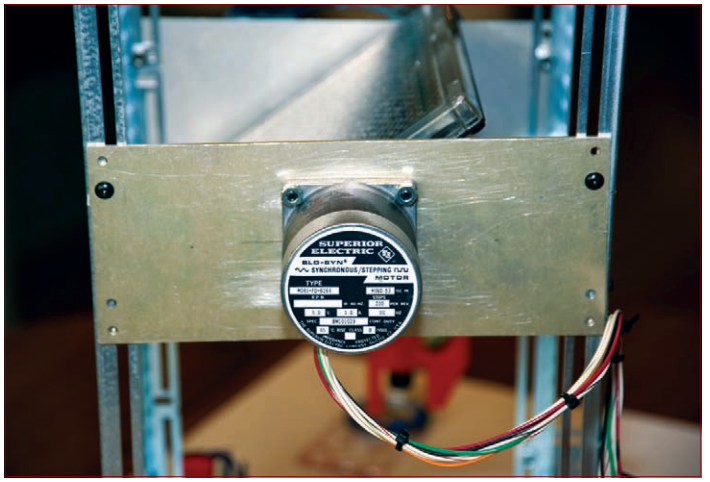


FIGURE 6. Shows the elevation drive stepper motor connected directly to the 12 volt solar panel with a 1/4 inch stainless steel rod. This motor is relatively heavy, so it needs to be counterbalanced.

The trim pot on each Easystepper board can be adjusted for current as described in the datasheets. You can also find the latest schematic for the SparkFun EasyDriver at **www.sparkfun.com/datasheets/Robotics/EasyDriver_v43.pdf**.

SunBot Calibration and Alignment

With your pair of protective goggles firmly in place, start the SunBot calibration process by collecting multiple sets of stepper counts, encoder counts, and optional counts from the encoder while the VEX microcontroller moves the panel to its extreme limits (0-360 degrees azimuth and ± 0-90 degrees elevation) using two protractors to take readings while the panels are being moved by the stepper motor drives. The data sets are then statistically averaged to take into account most of the sources of error so that those sources can be removed from the system. The results from this process are the minimum and maximum counts used to derive the calibration constants (gains and offsets) that are used to generate stepper motor commands for particular sun azimuth and elevation angles. In addition, the home position and hard mechanical limits are determined with VEX limit switches so that these limits (hard stops) are not exceeded. Don't worry. The process is relatively easy to do and we will provide all the necessary information to carry it out below. **Listing 1** shows some of the calibration code written in PIC18 C and WPILIB code. For the complete listing, go to the *SERVO* download web page for this article.

Stepper motors can be controlled in open loop fashion (no feedback required) if the motors have been properly calibrated. Otherwise, it is possible for them to misstep due to various conditions including too heavy a load, friction issues, and gear backlash. This calibration is very similar to the calibration performed by the internal firmware in printers and plotters and does require some feedback, usually from an encoder and limit switches that detect the hard mechanical stops.

The VEX quadrature optical encoders were useful during the calibration process to verify that the stepper motors did not skip any steps. Fortunately, it turns out that the larger direct drive steppers did not skip any steps, but unfortunately the 90 CPR resolution of the optical encoders was not precise enough for actually calibrating each axis, although they can be used as a rough approximation or even as a backup to the limit switches. Instead, for calibration, I used low cost protractors that I purchased at a local office supply store that allow us to measure angles between 0 and 180 degrees to within one degree resolution for the elevation axis, and a 0 to 360 degree protractor for the azimuth axis for use in the calibration process which is to move the SunBot from 0 to 360 degrees or 0 to 180 degrees azimuth while collecting azimuth encoder counts and stepper counts until it is stopped when it reaches the maximum angle of travel (180 or 360 degrees azimuth).

The elevation scale protractor on the elevation axis is used to determine how many steps it takes for the stepper motor to move from zero degrees elevation to 90 or 180 degrees elevation during the calibration process. I just mounted it on the elevation axis so that it turns with the stepper motor. It is measured against the fixed horizontal line denoting the x axis.

The azimuth axis scale (setting circle) is the enlarged ring shown in **Figure 1**, placed surrounding the entire the azimuth assembly. During calibration, it is used to determine how many stepper counts it takes to move the stepper motor from 0 to 360 degrees azimuth, with the laser pointer placed above it parallel to the elevation axis. The laser pointer should be placed on the elevation subassembly so that it is directly above the azimuth setting circle. The red laser spot is then used to read the azimuth angles in degrees. Repeat the same procedure for the elevation axis except stop the stepper motors at 90 degrees when the solar panel is exactly vertical. It is not recommend that you use hard stops to stop the stepper motors since they could be damaged. Instead, use the limit switches to stop the motors as shown in **Listing 1**.

This process is repeated 10 or 20 times for each axis and the data collected is averaged. The stepper counts should correlate to the encoder counts, although they won't be exactly 1:1 unless you use a high resolution encoder. For a directly driven solar panel, the results should be around 1,600 stepper counts to move 0 to 360 degrees azimuth and 400 stepper counts to move 0 to 90 degrees elevation. If you get counts that are much more or much less, then you may need to mechanically adjust or align the SunBot to make sure it is not sticking or binding on wires or metal parts.

Position Control

We can use the simple equation shown in **Figure 7** to scale the azimuth and elevation angles in degrees to actual stepper motor commands that are sent to the Easystepper driver boards.Once SunBot has been calibrated, moving a stepper motor to a particular position (orientation) is just a matter of toggling the STEP input to the selected Easystepper board for the desired number of steps. This is calculated using the equation in **Figure 7**.

```
/*************************************************
 * CalibrateAzimuthDrive -  This procedure
 * calibrates and aligns the Azimuth Stepper
 * Motor drive. It does this by finding 0
 * degrees Azimuth  (backside of  Azimuth limit
 * switch) and counts the number of steps
 * counterclockwise until the Azimuth Limit
 * Switch is triggered. It returns a TRUE if
 * successful.
 *************************************************/
int CalibrateAzimuthDrive (void)
{

int TotalNumberAzimuthSteps = 0;
    // Total number of steps corresponding to
    // 360 degrees

  PrintToScreen("trace 1: Start of Azimuth
  Drive Calibration \r");
    //PrintToScreen("1. Position the Sunbot
    // platform clockwise to 0 degrees Azimuth
    // (backside of  Azimuth limit switch).
    // \r");

    // Move The Azimuth Stepper Motors
    // CounterClockwise 1 step at a time while
    // checking the CCW Limit Switch state.
    // while(GetDigitalInput(AzimuthCWLimit)
    // == 1)
  while(GetDigitalInput(AzimuthCCWLimit) == 1)
  {
    TotalNumberAzimuthSteps++;
    //PrintToScreen("2. TotalNumberAzimuthSteps
    // =  %d \r", TotalNumberAzimuthSteps);
    // MoveStepper(AZIMUTH_STEPPER_MOTOR,     //
CLOCKWISE, 1);
    //Wait(100);

    // Get the latest encoder reading from the
    // Azimutn Drive
   AzimuthEncoderCounts = - GetQuadEncoder
   (AzimuthEncoderInt,  AzimuthEncoderInput);
   PrintToScreen("AZ Encoder Counts =  %ld\r",
   AzimuthEncoderCounts);
  }

PrintToScreen("2. TotalNumberAzimuthSteps =
%d \r", TotalNumberAzimuthSteps);
PrintToScreen("   AzimuthEncoderCounts     =
%ld \r", AzimuthEncoderCounts);

PrintToScreen("trace 3: End of Azimuth Drive
Calibration \r");
    return TRUE;
}
```

```
/*************************************************
 * CalibrateElevaionDrive -  This procedure  *
* calibrates and aligns the Elevaion Stepper
 * Motor drive. It does this by finding 0
 * degrees Elevation (backside of Elevation
 * limit switch) and counts the number of steps
 * counterclockwise until the Elevation Limit
 * Switch is triggered.
 *************************************************/
int CalibrateElevationDrive (void)
{
 int TotalNumberElevationSteps = 0;
    // Total number of steps corresponding to
    // 360 degrees

 PrintToScreen("trace 1: Start of Elevation
Drive Calibration \r");
    //PrintToScreen("1. Position the solar panel
    // platform clockwise to 0 degrees
    // Elevation (backside of Elevation limit
       switch).  \r");

    // Move The Elevation Stepper Motors
    // Clockwise 1 step at a time while
    // checking the CW Limit Switch state.
    // while(GetDigitalInput(ElevationCWLimit)
    // == 1)
  while(GetDigitalInput(ElevationCCWLimit) == 1)
  {
    TotalNumberElevationSteps++;
    // Total number of steps corresponding to
    // +/-90 degrees Elevation
   //PrintToScreen("2. TotalNumberElevationSteps
    // =  %d \r", TotalNumberElevationSteps);

      MoveStepper(ELEVATION_STEPPER_MOTOR,
      CLOCKWISE, 1);
    //Wait(100);

    // Get the latest encoder reading from the
    // Elevation Drive
   ElevationEncoderCounts = GetQuadEncoder
   (ElevationEncoderInt, ElevationEncoderInput);
   PrintToScreen("El Encoder Counts =  %ld\r",
   ElevationEncoderCounts);
   //Wait(100);
  }
  PrintToScreen("2. TotalNumberElevationSteps =
  %d \r", TotalNumberElevationSteps);
  PrintToScreen("   ElevationEncoderCounts
  =  %ld \r", AzimuthEncoderCounts);

  PrintToScreen("trace 3: End of Elevation Drive
   Calibration \r");

  return TRUE;
}
```

LISTING 1. A portion of the necessary calibration PIC18 C and WPILIB code using IEEE567 floating point arithmetic. For the complete listing, go to the *SERVO* downloads for this article.

Remember that most common stepper motors move 1.8 degrees per step, although some can move in either larger or smaller angles, depending on the make and application. This specification is usually printed on the motor along with the rated coil activation voltage and current (shown previously in **Figure 6**).

It is always a good idea to start moving stepper motors from a safe home position, usually placed between either extreme axis limit so that it does not accidently travel to the limit switches or hard stops and cause damage to the mechanism. The zero degree azimuth position (North) and the horizontal position of the elevation axis (zero degrees) are good locations for home, although these may be different for your applications. Azimuth and elevation angles are measured from the x axis or y axis, and can be verified by inspection using the protractor with degree markings. I like the ones with a ruler attached to the center that can be used for measuring the angles.

For example, in a perfect world with ideal motors the proper load (and no friction) to move the solar panel 30 degrees azimuth using full steps (1.8 degrees/step) for a total of 16 full steps (int(30/1.8) = 16) gives us 28.8 degrees of movement (16 x 1.8 = 28.8). The error is 30 – 28.8 = 1.2 degrees or 4% error margin.

If we use micro-stepping, the situation improves dramatically (as shown in this example). To move 30 degrees azimuth using 1.8/8 = 0.225 and int(30/0.225) = 133 gives us 29.925 degrees of movement (133 x 0.225). The error margin in this case is 30 – 29.925 = 0.075 or 0.25%. This result is even better than ± 1 degree that I originally specified for SunBot.

The above examples are for directly driven loads with no gear reduction. When you add gear boxes, the step size is further reduced depending on the gear ratio, but gear

backlash and friction can introduce other errors to the stepper motor movements. The calibration process mentioned previously helps to remove some of these errors.

Astronomy

WARNING: NEVER LOOK DIRECTLY AT THE SUN OR USE ANY OPTICS OR EVEN MIRRORS TO LOOK AT THE SUN! THIS KEEPS YOUR EYES SAFE FROM HARMFUL UV.

Knowing the sun's position at any time of day is useful to point the solar panel or reflector. This information is crucial for SunBot. You will need to use the magnetic compass in order to align the SunBot's zero degree azimuth with true North. An electronic compass could also be used to determine the direction but is not necessary for this application. The protractor mentioned previously will also help you to see if the SunBot has moved correctly. You will need to find your current geographic location (Latitude, Longitude); use a GPS to obtain them or get the same information from geographic maps of your location, or even find it on the Web.

There are many ways to derive the azimuth and altitude (elevation) angles for the sun for a particular time period. Let's look at two methods. The first method is to compute these angles using equations found on the Web at **http://en.wikipedia.org/wiki/Solar_azimuth_angle**. You could use Excel or MATLAB or even a C application running on a laptop to evaluate these sun azimuth and elevation equations. Examining these equations at these websites closely, you find that there are a few sine and cosine functions embedded in the math.

The second option (which is much easier) is to go to the Naval Observatory site and use their calculator which is freely available at **www.usno.navy.mil/USNO/astronomical-applications/data-services/alt-az-us**. This calculator provides a table that gives you the sun's azimuth and elevation for any time interval in any particular day.

Using the Naval Observatory, I obtained the sun positions for January 1st, 2010 as shown in **Table 2**. The table shows the sun's azimuth and elevation (altitude) every 10 minutes for Milford, NH. These same tables can be used to determine one's current position anywhere on the earth's surface by doing the exact opposite operation and measuring the sun's current position, then obtaining the current Universal Time (UT) or Zulu Time (Greenwich England) to obtain the geodetic latitude and longitude coordinates. This is the way navigation was accomplished before GPS by ship's navigator using an astrolabe to obtain the sun's altitude and a compass using dead reckoning techniques. It's still used as a backup in case the GPS stops working. These navigation equations use spherical trigonometry involving some sine and cosine coordinate transformations, and even use the same pi constant (accurate to many more decimal places) that our friend Archimedes helped to calculate over 2,000 years ago.

The astronomical terms used in the table are described as follows: ***Altitude (elevation)*** is the angle up from the horizon. Zero degrees altitude means exactly on your local horizon, and 90 degrees is straight up. ***Azimuth*** is the angle along the horizon, with zero degrees corresponding to North, and increasing in a clockwise fashion. Thus, 90 degrees is East, 180 degrees is South, and 270 degrees is West. Using these two angles, one can describe the apparent position of an object (such as the sun) at a given time. The altitude and azimuth values are for the center of the apparent disk of the sun or moon.

Stepper Motor Controller Firmware

A PIC18 C application rotates the SunBot III azimuth axis from 0 to 90, to 180 to 360 degrees (or until the limit switch is triggered), then back to its home position of zero degrees (as shown in **Listing 2**).

Clocking the stepper motors at various rates can be accomplished by the VEX microcontroller using PIC18 C to generate an interrupt from one of its timers to clock the stepper motor at a specific rate (steps/second). With Easy C professional and WPILIB, we can use the Wait(n) statement inside a loop to clock the stepper motor using the specified period, but ISRs (Interrupt Service Routines) are not allowed. Although, there is the interrupt watcher function that can watch for specific interrupts such as the timer interrupts. Another approach is to use the WPILIB Register Repeating Timer function as shown in **Listing 3**. For VEX users who use EasyC Professional, the task of Porting this application is made easier because the WPILIB function calls are the same. The complete source code for the SunBot is available for download from the *SERVO* site.

How It Performed

SunBot III worked much better than the geared SunBot II version described last time. It was able to position the solar panel to various orientations very accurately using the micro-stepping mode without misstepping. For example, we put it through an azimuth position test after it was calibrated that moved it from zero degrees to 90 degrees to 180 degrees to 270 degrees, and back to zero degrees multiple times. The laser pointer spot landed exactly on each of these positions on the azimuth setting circle. The same was true moving the elevation axis from zero to 45 degrees to 90 degrees, and back to zero degrees. Using the astronomical calculators, you should be able to have it point directly at the sun or moon. Other objects in the sky require even more pointing precision than this design can provide, although it could be a starting point for a DIY telescope.

This design has a problem that the previous geared version did not, and that is it loses its position when powered off. Another problem is that stepper motors are a bit noisy and generate vibrations when they move; they even vibrate slightly when stationary. This causes problems in applications such as security monitoring, astrophotography, or nature photography. A possible

FIGURE 7. This simple equation is used to calculate the number of steps required to move the 1.8 degree per step stepper motor to any angle between zero and 360 degrees using 1/8 micro-steps. N is the number of steps sent to the Easystepper board and Angle is the azimuth or elevation angle of the sun in degrees.

```
N = int(Angle/0.225)
```

Astronomical Applications Dept.
US Naval Observatory
Washington, DC 20392-5420
MILFORD, NEW HAMPSHIRE
o,o,
W 71 39, N42 50
Altitude and Azimuth of the Sun
Jan 1, 2011 Eastern Standard Time

H:M	Altitude°	Azimuth (E of N)°
06:10	-11.9	110.4
06:20	-10.2	112.0
06:30	-8.5	113.6
06:40	-6.9	115.2
06:50	-5.2	116.8
07:00	-3.6	118.4
07:10	-2.0	120.1
07:20	0.1	121.7
07:30	1.5	123.4
07:40	2.9	125.2
07:50	4.3	126.9
08:00	5.7	128.7
08:10	7.1	130.6
08:20	8.5	132.4
08:30	9.8	134.3
08:40	11.1	136.3
08:50	12.3	138.2
09:00	13.5	140.2
09:10	14.6	142.3
09:20	15.7	144.4
09:30	16.7	146.5
09:40	17.7	148.7
09:50	18.6	150.9
10:00	19.5	153.2
10:10	20.3	155.5
10:20	21.0	157.8
10:30	21.7	160.2
10:40	22.3	162.6
10:50	22.8	165.0
11:00	23.2	167.5
11:10	23.6	169.9
11:20	23.8	172.4
11:30	24.1	174.9
11:40	24.2	177.4
11:50	24.2	180.0
12:00	24.2	182.5
12:10	24.1	185.0
12:20	23.9	187.5
12:30	23.6	190.0
12:40	23.2	192.5
12:50	22.8	194.9
13:00	22.3	197.4
13:10	21.7	199.8
13:20	21.0	202.1
13:30	20.3	204.5
13:40	19.5	206.8
13:50	18.7	209.0
14:00	17.8	211.2
14:10	16.8	213.4
14:20	15.8	215.6
14:30	14.7	217.7
14:40	13.5	219.7
14:50	12.3	221.7
15:00	11.1	223.7
15:10	9.8	225.7
15:20	8.5	227.6
15:30	7.1	229.4
15:40	5.8	231.2
15:50	4.4	233.0
16:00	2.9	234.8
16:10	1.5	236.5
16:20	0.2	238.2
16:30	-1.9	239.9
16:40	-3.5	241.6
16:50	-5.2	243.2
17:00	-6.8	244.8
17:10	-8.5	246.4
17:20	10.2	248.0
17:30	-11.9	249.6

TABLE 2. Using the Naval Observatory calculator, I obtained the sun's positions for January 1st, 2010. The table shows the sun's azimuth and elevation (altitude) every 10 minutes for Milford, NH.

solution to these problems is to devise some kind of electromagnetic brake or even to use the lead screw invented by Archimedes to drive a stepper motor for the elevation axis.

Other areas that need to be improved to make SunBot III more practical include adding a real time clock to the microcontroller so that it knows when to move the solar panel. Another area is communication so that it can read the positions from a laptop host using the serial port. Finally, the floating point processing is a big issue since the VEX microcontroller v 0.5 runs out of memory quickly when processing a lot of floating point equations. For this reason, a host-based application such as Excel or MATLAB or even C++ may be needed to convert the angles to stepper motor commands before they are sent to the microcontroller.

Advanced Stepper Motor Control

One major source of error during open loop stepper motor control is that a stepper motor can slip one or more steps if overdriven, thus causing errors in position and speed during normal operation. To mitigate this problem, stepper motor commands should be ramped to a final position or velocity using motion profiles (ramp, trapezoid, etc.) for a smooth transition. In addition, a current feedback PID control loop could be used to control the stepper motor's position or speed, using a quadrature optical encoder for feedback. In fact, Microchip provides an application note (AN1307 - Stepper Motor Control with dsPIC; **www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1824&appnote=en546027**) for using an advanced dsPIC that provides precise position and speed control for stepper motors with micro-stepping up to 1/64 of a step. A dsPIC is just a 16-bit PIC with some Digital Signal Processing (DSP) hardware, whereas the VEX microcontroller uses the Microchip eight-bit PIC18F8520.

Other SunBot Applications

Other applications for SunBot include using the

```
PrintToScreen("Moving Azimuth stepper
motor...\r");

// Move The Azimuth Stepper Motors.  Works!!!
while (1)
{
   // Move 90 degrees Azimuth
 MoveStepper(AZIMUTH_STEPPER_MOTOR, CLOCKWISE,
 400);
 Wait(2000);
 MoveStepper(AZIMUTH_STEPPER_MOTOR,
 COUNTER_CLOCKWISE, 0);
 Wait(2000);

   // Move 180 degrees Azimuth
 MoveStepper(AZIMUTH_STEPPER_MOTOR, CLOCKWISE,
 800);
 Wait(2000);
 MoveStepper(AZIMUTH_STEPPER_MOTOR,
 COUNTER_CLOCKWISE, 0);
 Wait(2000);
   // Move 270 degrees Azimuth
MoveStepper(AZIMUTH_STEPPER_MOTOR, CLOCKWISE,
1200);
Wait(2000);
MoveStepper(AZIMUTH_STEPPER_MOTOR,
COUNTER_CLOCKWISE, 0);
\Wait(2000);

\
   \// Move 360 degrees Azimuth
MoveStepper(AZIMUTH_STEPPER_MOTOR, CLOCKWISE,
1600);
 Wait(2000);
MoveStepper(AZIMUTH_STEPPER_MOTOR,
COUNTER_CLOCKWISE, 0);
Wait(2000);

}
```

LISTING 2. This PIC18 C application rotates the SunBot III azimuth axis from zero to 90 to 180 to 360 degrees (or until the limit switch is triggered), and then back to its home position of zero degrees.

```
   // Initialize repeating timers used to
   // update stepper motors every n
   // milliseconds using a repeating timer
   RegisterRepeatingTimer(1000,
   UpdateSteppers);

   while(1)
   {
      //PrintToScreen("TestRun\r\r");
   }
```

LISTING 3. This demonstrates how each Easystepper board is clocked by calling the UpdateSteppers function every second using the WPILIB Register Repeating Timer function.

tilt/pan frame as a low cost camera mount for security applications, solar power applications (heliostats), or telescope/binocular mount for taking pictures of near space objects such as the moon and planets. The moon in particular is easier to track (like the sun) with a homemade tracker since it is a visually larger disk. The moon's nightly position is also available from astronomical calculators on the Web.

Conclusion

In this article, we described how to build a simpler version of SunBot than the geared version described previously. We used the VEX microcontroller to control large stepper motors used to directly drive the solar panel in both azimuth and elevation. We also showed you how to read VEX limit switches so that SunBot does not wrap the cables around each axis, and how to calibrate the stepper motors using quadrature encoders and a low cost protractor to be able to position the solar panel 0-360 degrees azimuth and ±90 degrees elevation. In addition, we described the PIC18 C and WPILIB firmware used to control the SparkFun Easystepper board.

Although SunBot III is a relatively simple VEX-based application, it has many expansion possibilities. Hopefully, you will be able to use the information presented here as a starting point for more advanced stepper motor applications, especially in astronomy. Finally, we showed you how to obtain the positions of the sun and moon using free astronomical calculators on the Web. Next time, I plan to show you how to use various sensors, including high resolution quadrature encoders and many other sensors that are not yet part of the VEX inventory. SV

THE ORIGINAL SINCE 1994

PCB-POOL®.COM

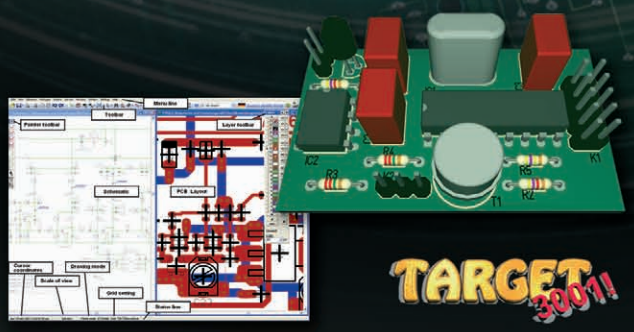
Beta LAYOUT

LOW COST PCB Prototypes

- Easy Online Quotations & Ordering
- No Pre registration
- No Tooling Charges
- No Minimum Quantity
- Single sided - 6 layers
- Lead times from 48 hours
- Full DRC CHECKS (included)
- Live Online Support

Need Layout Software?
Download Ours... its FREE!

- Schematic Capture
- PCB Layout
- Auto Router & Auto Placer
- Complete Component Library
- Professional Version



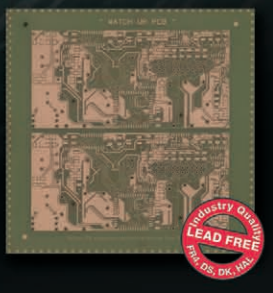
www.free-pcb-software.com

WATCH UR PCB®

View your PCB as it gets manufactured!

We send HD images at each production stage so you can track its progress!!

- Drilling
- Exposure
- Tin Stripping
- UV curing
- Hot air levelling (HAL)



www.pcb-pool.com

Email: sales@pcb-pool.com **Toll Free : 1877 390 8541**

Simply send your files & order ONLINE **WWW.PCB-POOL.COM**

Accepted file formats:

p-cad TARGET Protel EDWin orcad GraphiCode PROTEUS Electronics Workbench Easy-PC Sprint Layout GERBER

Beta LAYOUT

CPLDs — Part 2

complex programmable logic devices

Graphical Programming of a CPLD

by David A. Ward

The first article in this series introduced CPLDs (Complex Programmable Logic Devices) and listed some items that needed to be purchased before experimenting with them. If you have everything ready, we can now look at programming, breadboarding, and testing CPLDs. We'll be doing things a little out of order, since normally after designing a circuit you would run a simulation to see if it behaves the way you want. Then, you'd proceed to program the CPLD. The setup and running of a simulation will be saved for Part 3. We'll get right into programming at this time. A word of caution, however, concerning the steps we'll be going through in Xilinx ISE to get the CPLD programmed. The Xilinx ISE Project Navigator software can appear very complicated and many of the menus are context-sensitive. So, it's likely that you'll run into some difficulties, but it does work well and with a little time you won't feel so overwhelmed by everything going on in the Xilinx ISE windows.

When you launch Xilinx ISE 12.3, you'll see a tip of the day window. After this is closed, you can select the new project button on the lower left-hand side of the screen; it's located on the start tab (see **Figure 1**). Next, you'll see the New Project Wizard window as shown in **Figure 2**. Locate a directory and name your project. At the bottom of this window, you can select your Top-level source type — select schematic this time, and finally select next. The next window is the project settings window shown in **Figure 3**. For this project, we'll select an XC9572XL CPLD in a PC44 package with a speed of -10 nS. All of the other choices in this window should be okay in their default settings; then select next.

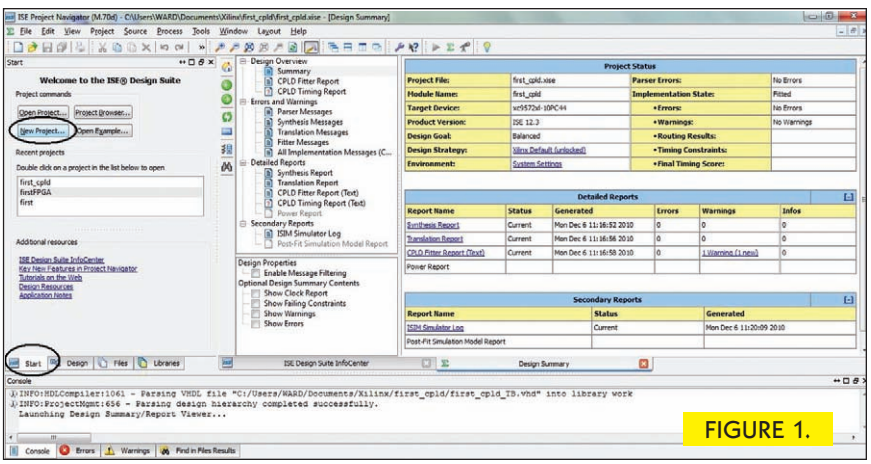


FIGURE 1.

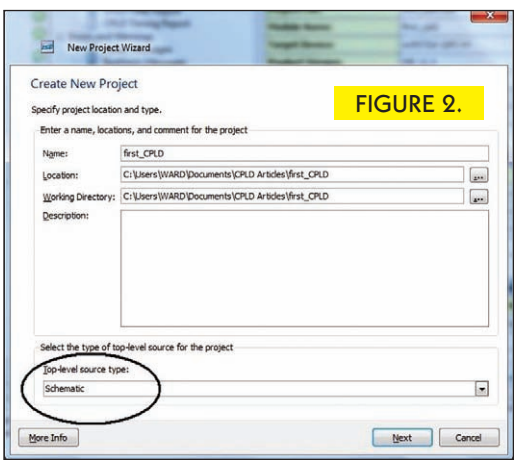
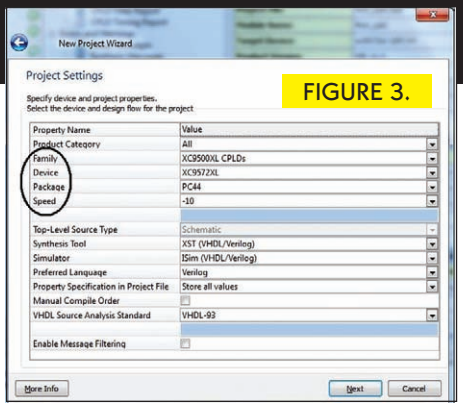
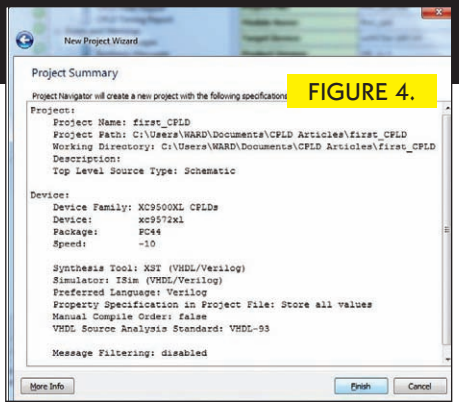
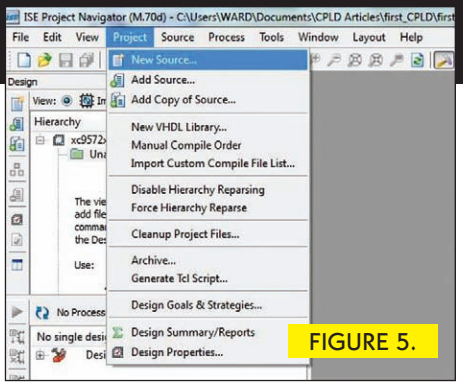
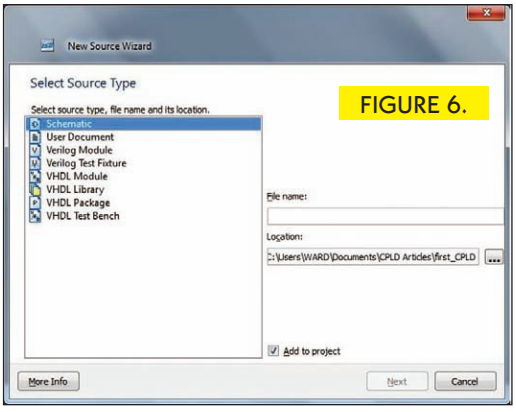


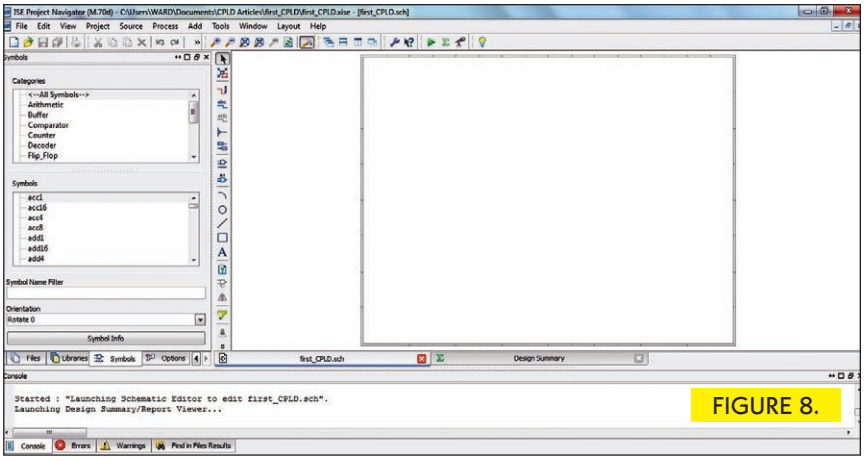
FIGURE 2.

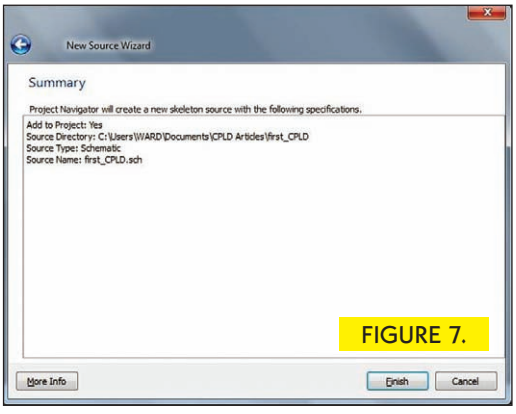


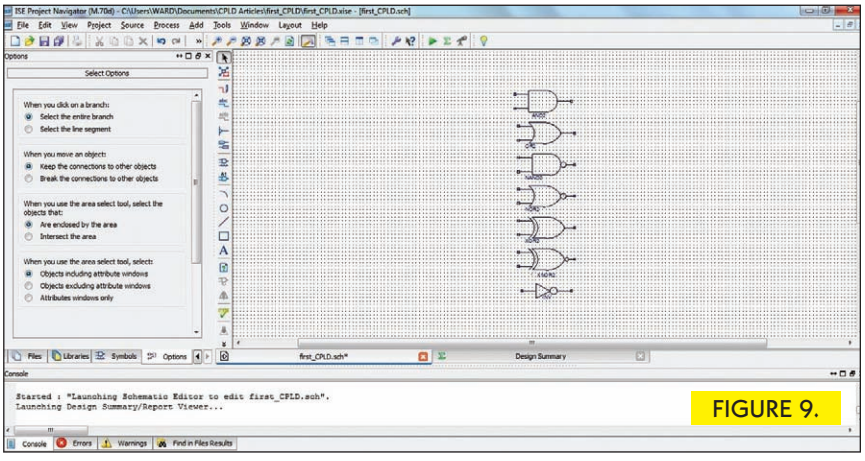




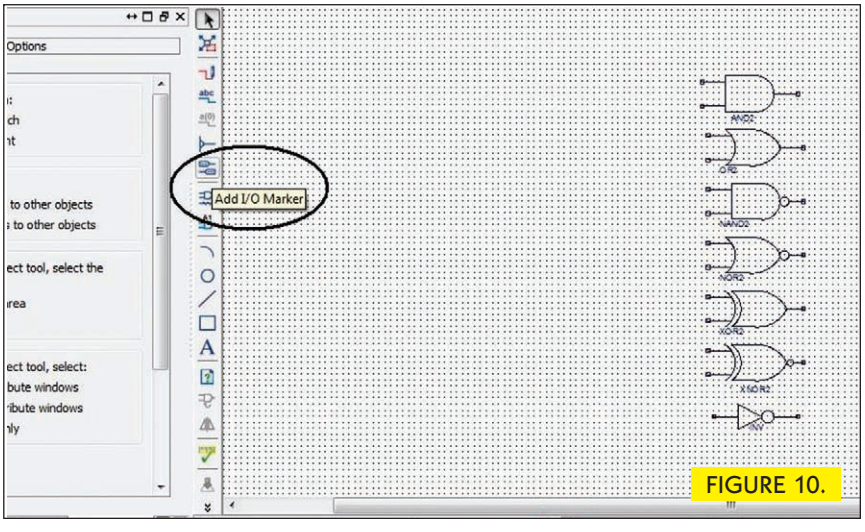








You'll now see a Project summary window (**Figure 4**); select finish. Next is the ISE project navigator window in **Figure 5**. From the top menu bar, select project and new source. Now, you are taken to the new source wizard window shown in **Figure 6**. Select schematic again and enter a filename and location for your new schematic source file; then select next. Now, you'll see the summary window as shown in **Figure 7**; select finish from here. You should now get a blank schematic page to enter your schematic diagram onto (**Figure 8**). With the symbols tab selected, you can place your logic gates down on the schematic page (**Figure 9**). After you have placed your desired logic gates down, select Add I/O Marker from the schematic tools on the left-hand side of the page. Place two input



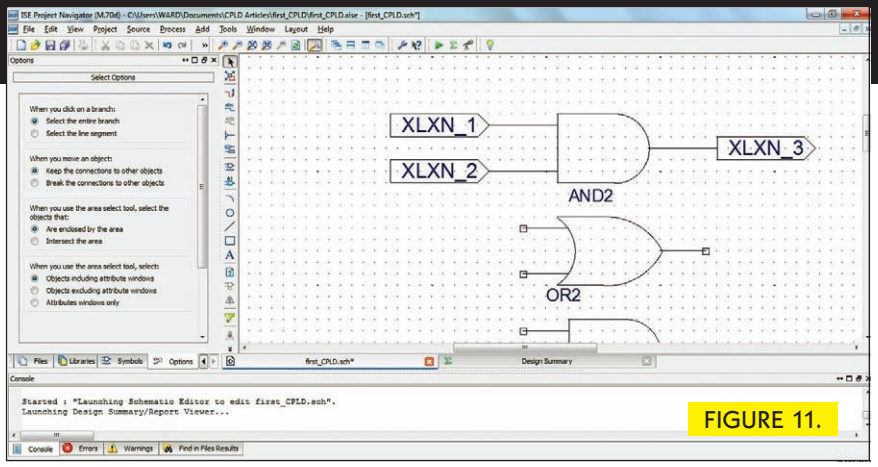


FIGURE 11.

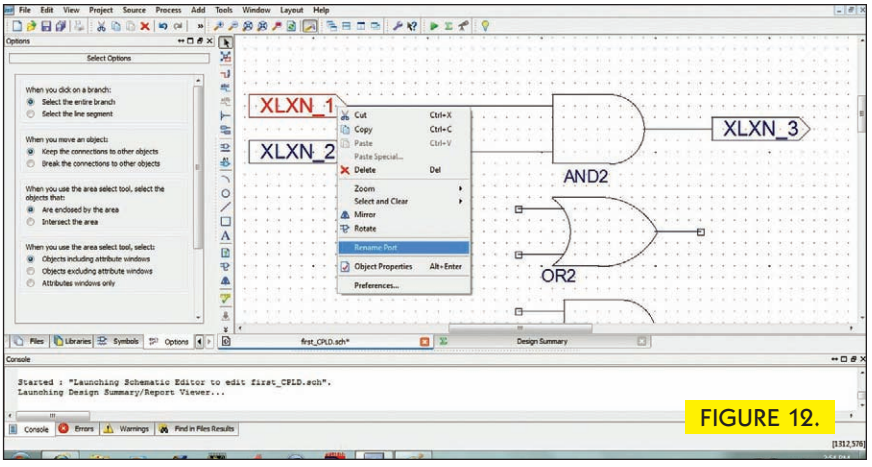


FIGURE 12.

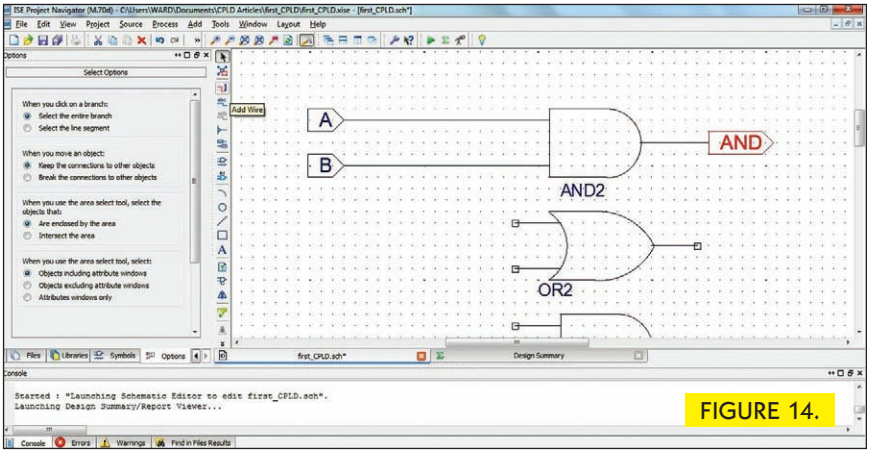


FIGURE 14.

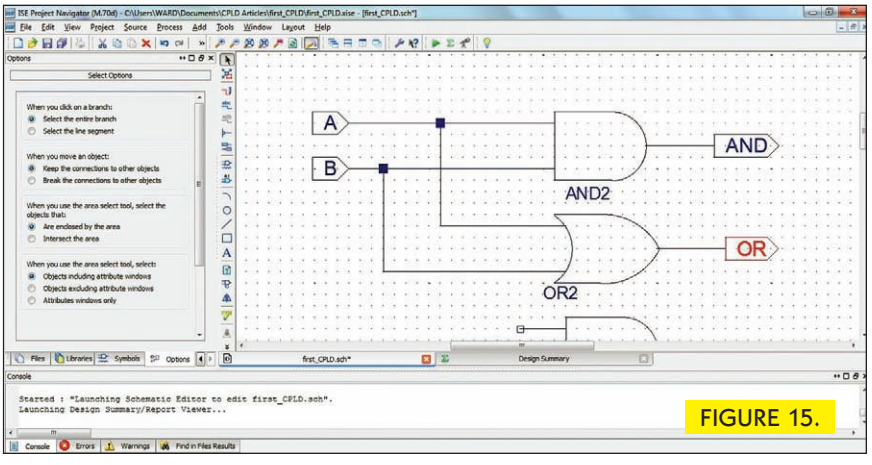


FIGURE 15.

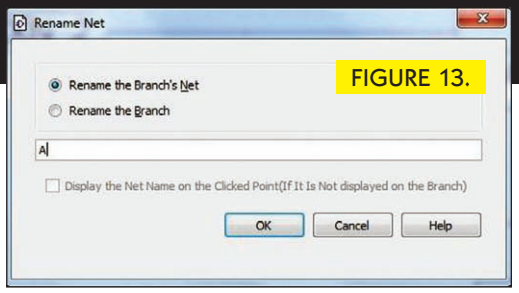


FIGURE 13.

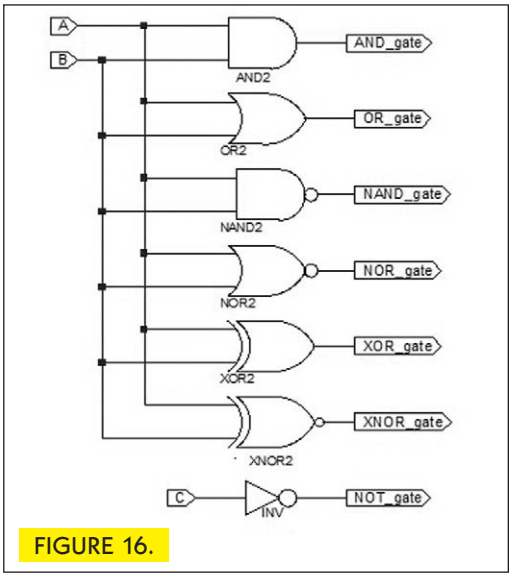


FIGURE 16.

markers on the left of the AND gate and one output marker on the right side (see **Figures 10** and **11**). The program knows when pins are inputs or outputs automatically.

You can experiment with all of the schematic drawing tools on your own later; we'll just explain the essentials here.

Next, let's rename the I/O markers. Left-click on a marker, then right-click for a drop-down menu and select rename port (**Figure 12**). The cursor needs to be the select cursor or arrow to do this. Type in a new name for the port (**Figure 13**). Next, drag and stretch the input ports out to the left to allow room for wires (**Figure 14**). Using the add wire tool, wire the gates together as in **Figure 15**. (It looks like a pencil drawing a red line.)

Our completed example circuit is shown in **Figure 16**; this circuit will incorporate all of the basic logic gates. Now, we are ready to compile the schematic diagram. Select implement top module from the top menu bar; it looks like a green run or play button (**Figure 17**). The program will churn for a while and it may not appear as if anything is happening. You'll see messages in the console window at the bottom as it works. If you have the design tab on the left-hand side of the screen open, you'll also see spinning blue circles as it works.

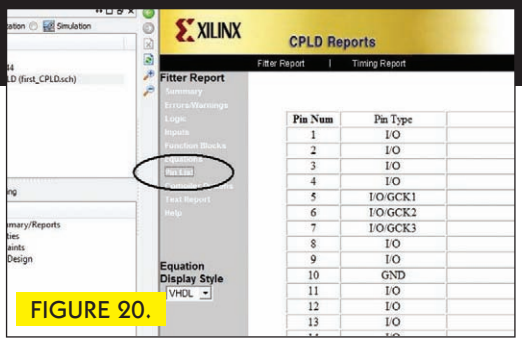


FIGURE 20.

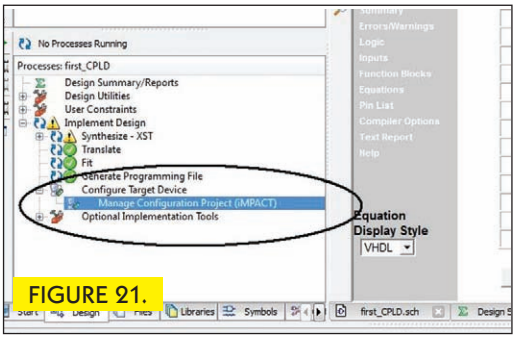


FIGURE 21.

If your circuit has any errors, you'll see a message in the console window at the bottom explaining what the error is (**Figure 18**). If everything is okay, you'll see the Xilinx CPLD report window showing how it fit your design into the CPLD package (**Figure 19**). From the CPLD report, select pin list on the left-hand side to see where the program placed your inputs and outputs in the CPLD itself (**Figure 20**).

If your design compiled okay, you're ready to program your CPLD; this occurs in another Xilinx program called "Impact." On the left-hand side of the screen with the design tab selected, expand the Configure Target Device icon, left-click, and then right-click on Manage Configuration Device Project (Impact) and select run (**Figures 21** and **22**). By the way, these menus are context-sensitive and if your schematic circuit is not selected in the design hierarchy view window, you may not see what is being described here.

When Impact runs, you will be taken into the Impact program (**Figure 23**). From this window, select File>New Project as shown in **Figure 23**. Next, you will see the window shown in **Figure 24**; select yes. The next window will connect and communicate with your JTAG programmer, so don't select OK until the programmer is connected to the computer and the CPLD has power connected to it (**Figure 25**).

If the communication process is successful, you'll see the window in **Figure 26**; select yes from this window. The next window (**Figure 27**) will have you find your compiled program that you want to program

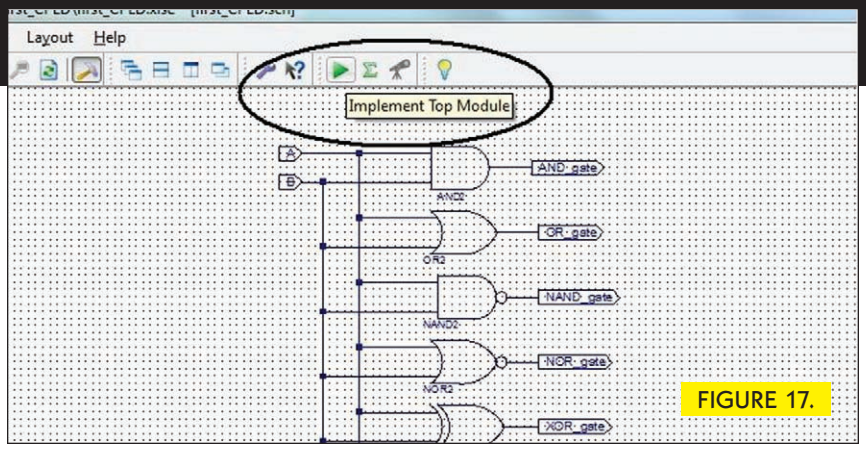


FIGURE 17.

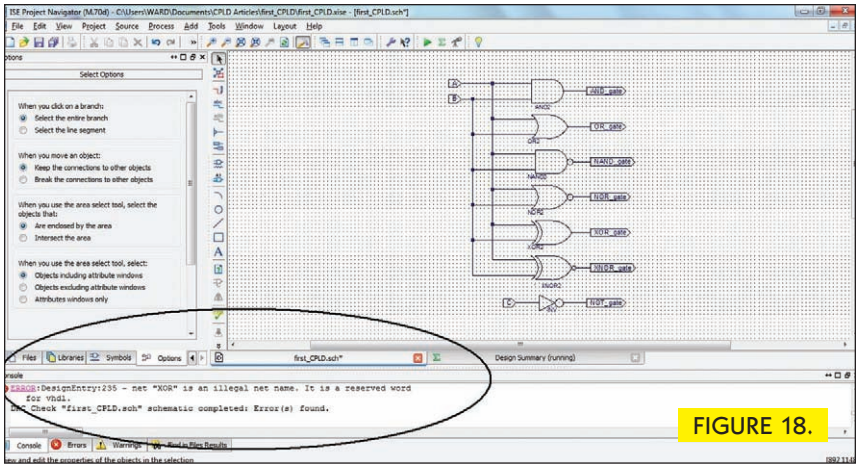


FIGURE 18.

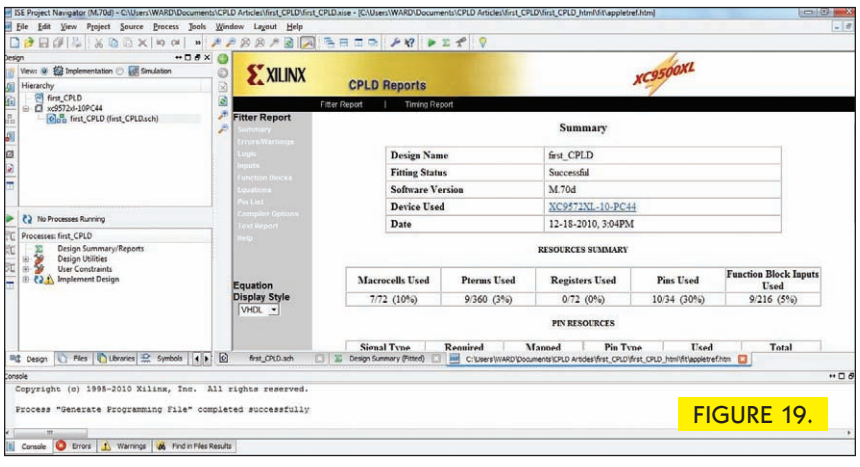


FIGURE 19.

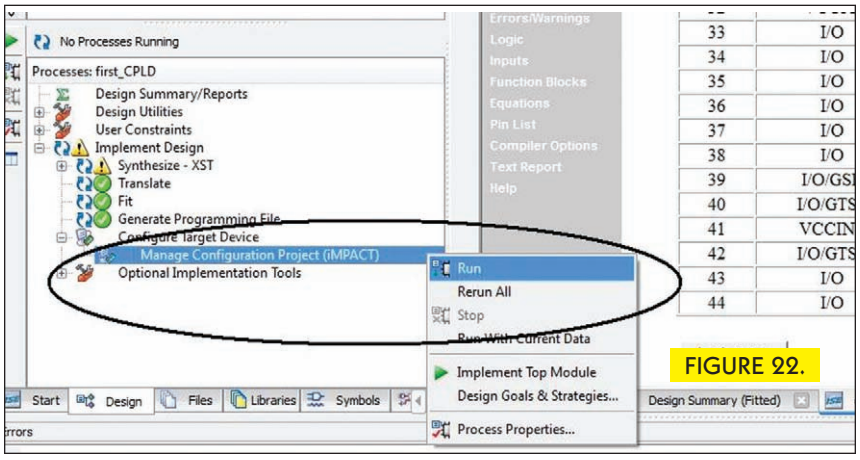


FIGURE 22.

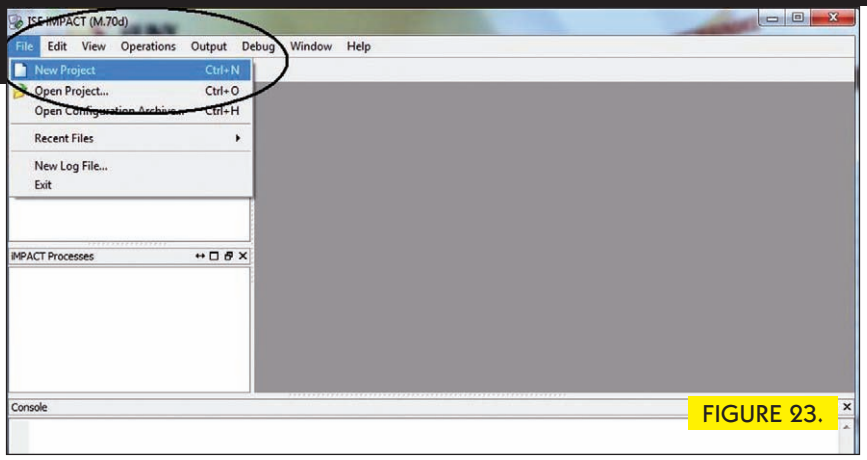


FIGURE 23.

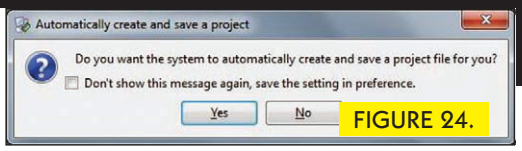


FIGURE 24.

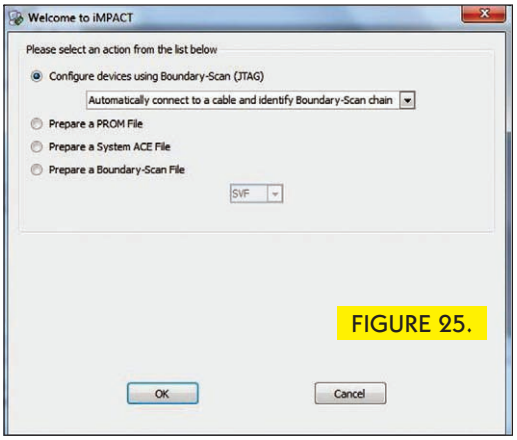


FIGURE 25.

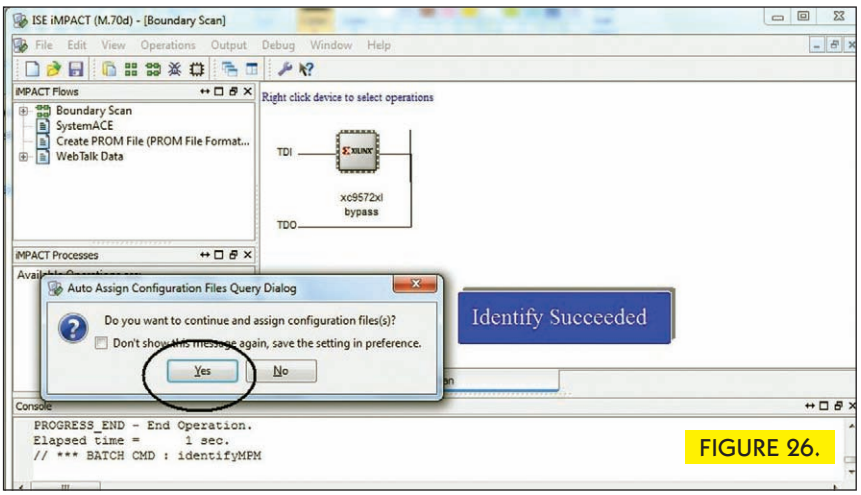


FIGURE 26.

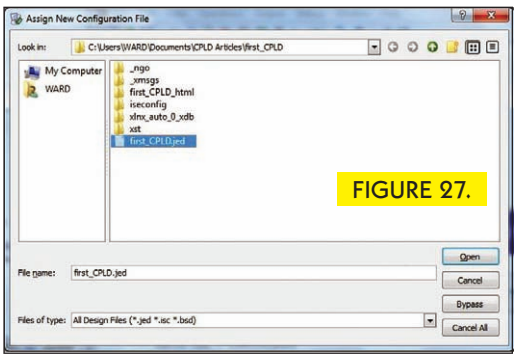


FIGURE 27.

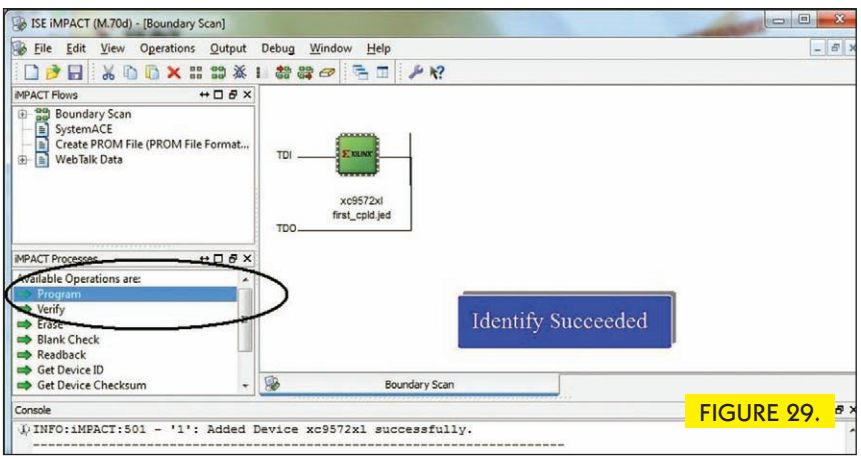


FIGURE 29.

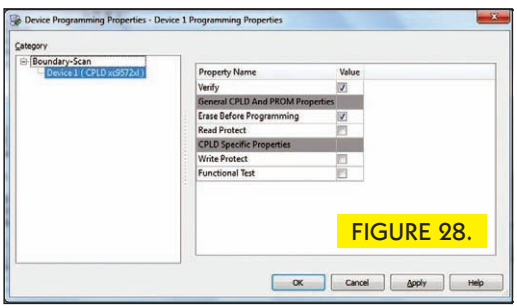


FIGURE 28.

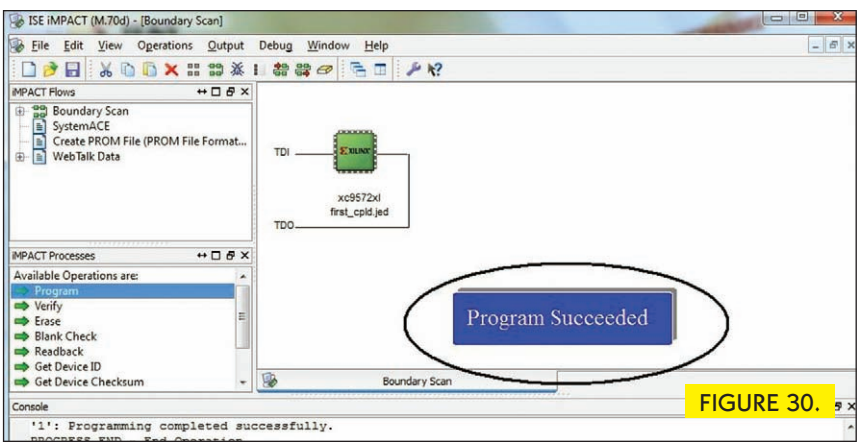
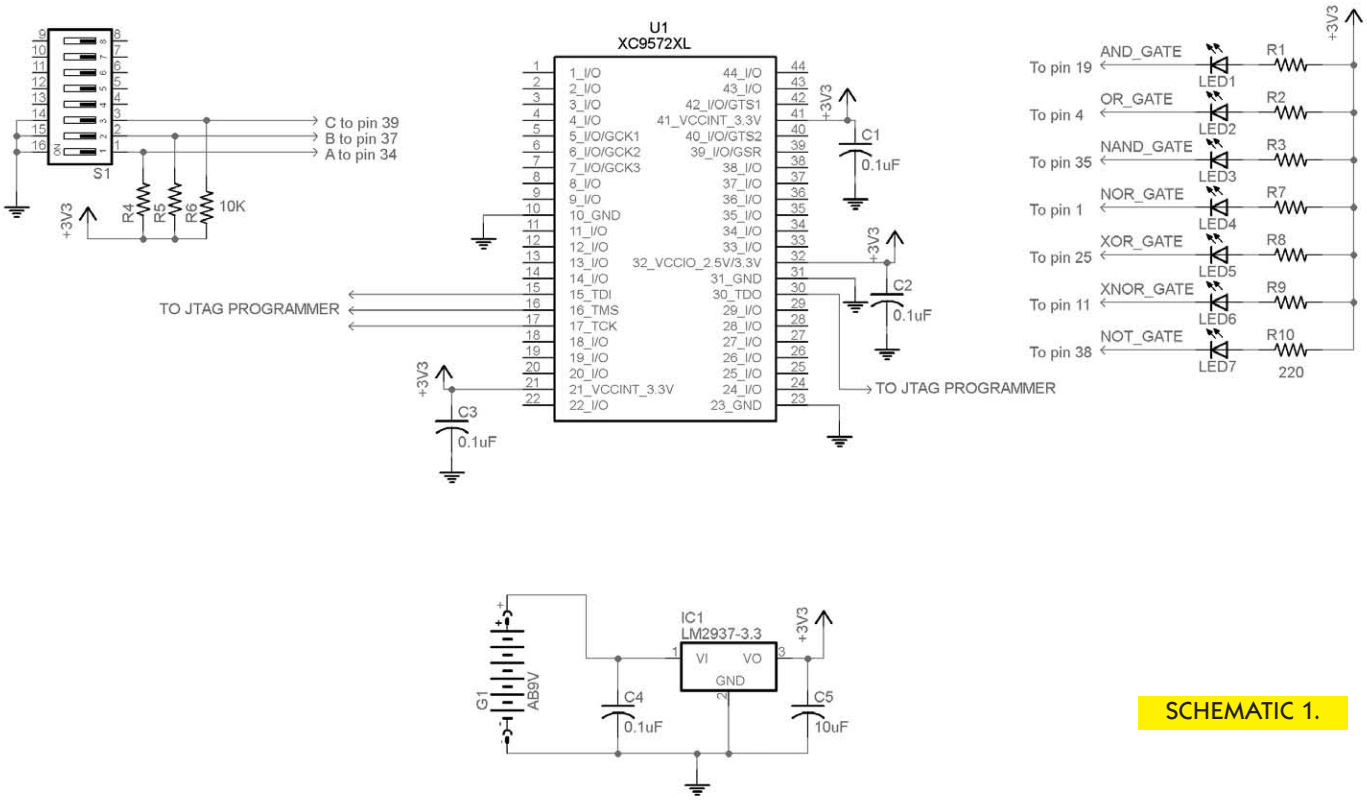


FIGURE 30.

into your CPLD. It has a .jed file extension; select open at this time. Select OK from the next window (**Figure 28**). Select program from the left-hand side of the window to actually program the CPLD (**Figure 29**). If all goes well, you'll see the Program Succeeded message in **Figure 30**. Whew, that wasn't so bad now, was it?

Now, you can test out the actual CPLD in circuit and see if it behaves the way it should. **Schematic 1** shows how to bread-board the CPLD for testing; also see **Photo 1**. Note that it is possible that your program may assign other pins for the inputs and outputs. Also note that if you make any changes to the schematic in Xilinx ISE and recompile the schematic, it may change the pin assignments. The program does have the



SCHEMATIC 1.

ability to lock the pins so they won't change from one compiling to the next. You can pre-assign pins, as well. This will be demonstrated in the next article. However, Xilinx recommends that you let the program determine the pin assignments since it will know how to best use the chip's internal structure in the most efficient manner. Going back to the schematic diagram, notice the cathodes of the LEDs are connected to the CPLD so that a 1 from the CPLD will turn the LED off and a 0 will turn the LED on. Xilinx datasheets recommend that LEDs be sunk rather than sourced by the CPLD and that the XC9572XL has a current drive strength of 8 mA sinking; see the XAPP805 application note. Note also that the resistors connected to the DIP switches are configured as pull-ups rather than pull-downs; this is another Xilinx recommendation.

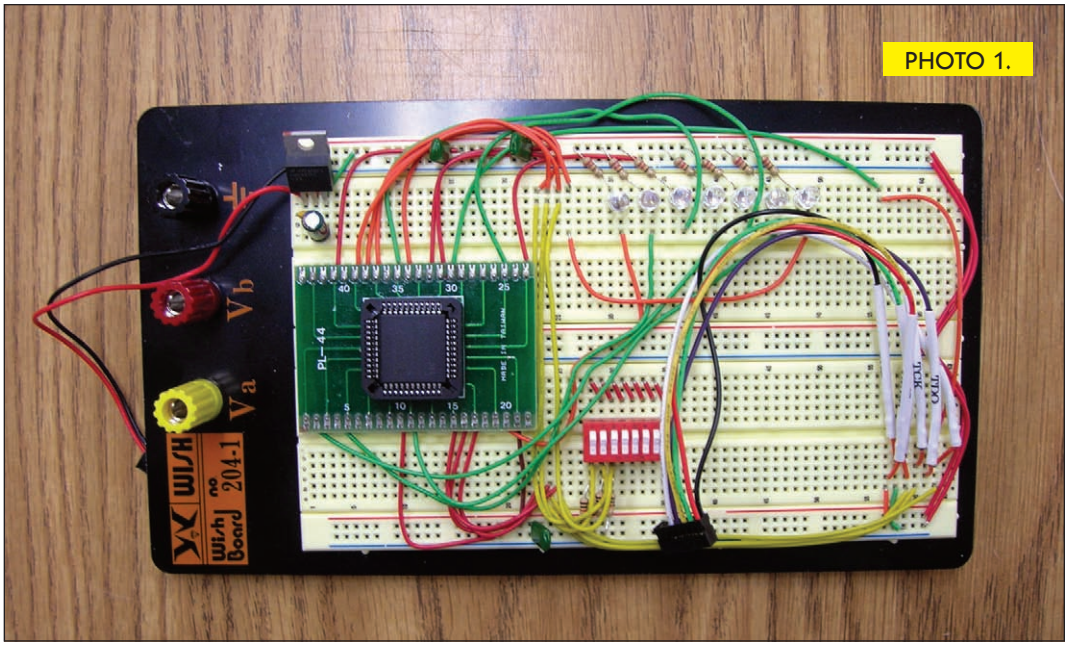


PHOTO 1.

It should be mentioned that the graphical method of programming CPLDs and FPGAs is really not the preferred method used by those in this field. Most serious programmers use HDL which is a text-based hardware description language. When circuits get more complicated, the HDL method can be a much simpler and more straightforward method than using the graphical method. I believe the graphical method is a quicker and easier way to get started. Next month, I'll demonstrate how to setup and run a simulation in Xilinx's ISIM program. We will need to generate an HDL test bench circuit in order to simulate the circuit in ISIM. SV

www.servomagazine.com/index.php?/magazine/article/april2011_Ward

Give Your Robot the Bootloader

by Fred Eady

I'm a 1950's kid. Thus, my first encounter with a robot was via television. If a robot didn't look like and act like Robby, it wasn't a robot. Later in my elementary school days, Robby was replaced by Robot B9 of the *Lost in Space* series. It is speculated that Robot B9 was also known as GUNTER which is an acronym for General Utility Non-Theorizing Environmental ROBOT. As movie and television robots became more cerebral, I remember going on a ninth grade science club road trip to see HAL in the movie *2001, A Space Odyssey*. I actually fainted in the theater.

As the years passed, the robots amassed. Good old *Star Trek* offered up every kind of robot that one could imagine, ranging from the beautiful bevy of Harcourt Fenton Mudd's Women and the android copy of his nagging wife Stella, to the Borg. The robots that were missed in the *Star Trek* series were realized in the *Star Wars* chronicles.

Here's a question for you. How many QWERTY keyboards can you remember seeing being used to update the firmware in robots like Robby or B9? How about, NONE! Recall that Star Wars' R2 units routinely attached themselves to data access portals on star cruisers and star bases. To me, that implies that any type of initial R2-D2 firmware loads came by way of a process we currently call bootloading.

As a child, I saw robots as humanoid programmable electromechanical devices with super human strength. As an adult, my robots are television set-top boxes, network routers, personal computers, cameras, telephones, and the gadgets that you and I build. The common factor in all of my "robots" is their ability to update the way they work without having to be reprogrammed by a human with a physical device programmer. For instance, from time to time the satellite box that services my television informs me that it downloaded a new feature or a program fix via the satellite link. The Canon 7D that I use to capture the photos for our *SERVO* discussions can also be fixed or updated via an image that I can transfer to the camera using the USB portal on my laptop. My Canon and my satellite box have one thing in common: They are endowed with some sort of bootloader core.

Not every device capable of accepting fixes and features via the bootloading process needs to have a satellite, USB, or Ethernet connection. We can use proven "it ain't dead yet" RS-232 techniques to load executable images into the robotic devices that you and I create.

Bootloaders From a Robotic Perspective

As a robot head, bootloading provides yet another layer of control for you. Once the base bootloader code is loaded into the robotic device's program Flash, you have the ability to read, write, and erase the program Flash without having to use a hardware programmer. Since you have privy to the microcontroller's program Flash, you also have programmatic access to all of the microcontroller's resources which include the microcontroller's EEPROM, SRAM, and configuration memory area.

To utilize the bootload process, the target microcontroller must be able to store the incoming image in a buffer for later programming if the currently running application cannot be interrupted. If the resident application can be interrupted and the new incoming bootload image data is coming in faster than the microcontroller can write it to Flash, the microcontroller is still required to be able to store the bootload image in a buffer.

Today's microcontrollers can do some incredible things. However, they can't compute on their own. Thus, a bootloader program must be initially loaded into the microcontroller's program Flash manually. The core bootloader code is loaded and positioned as to allow the bootloader application to always reside in a protected portion of the microcontroller's program Flash. Normally, the entry point to the core bootloader program is loaded at the microcontroller's execution start point. This allows the bootloader program to run at every microcontroller reset. It

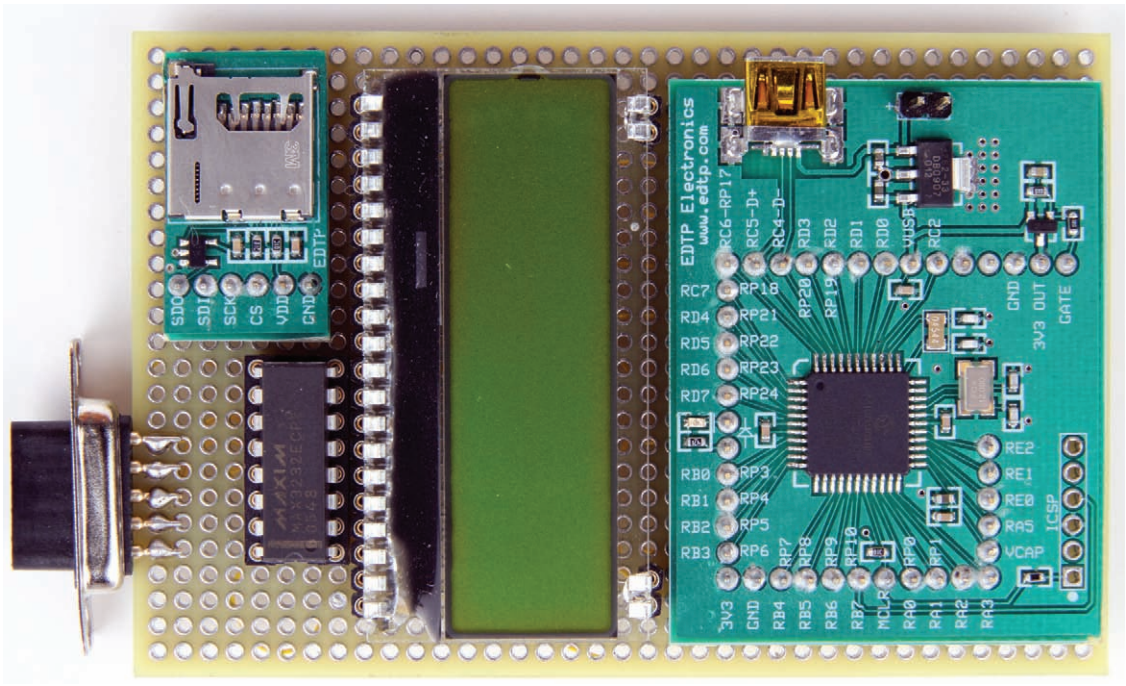


PHOTO 1. The PIC18F47J53 is a versatile microcontroller. Thus far, we've hung an SPI-based LCD on it. We've attached a microSD card to it and it has performed as a USB device. We're about to call upon its RS-232 talents.

also allows the bootloader program to determine if it needs to be bypassed and cede program control to the application that is currently residing at a predetermined location in the program Flash.

Depending on the microcontroller architecture, the actual bootload code segment can be located anywhere in program memory. However, you'll normally find the core bootloader code at the beginning or at the end of program Flash. The advantage of placing the bootloader code at the end of program memory lies in the fact that you don't have to relocate the microcontroller's interrupt vectors. However, if you choose to locate the bootloader code at the beginning of Flash, it's not a big deal to redirect the microcontroller's interrupt vectors. With that, let's put together a hardware platform with a rear-mounted bootloader core.

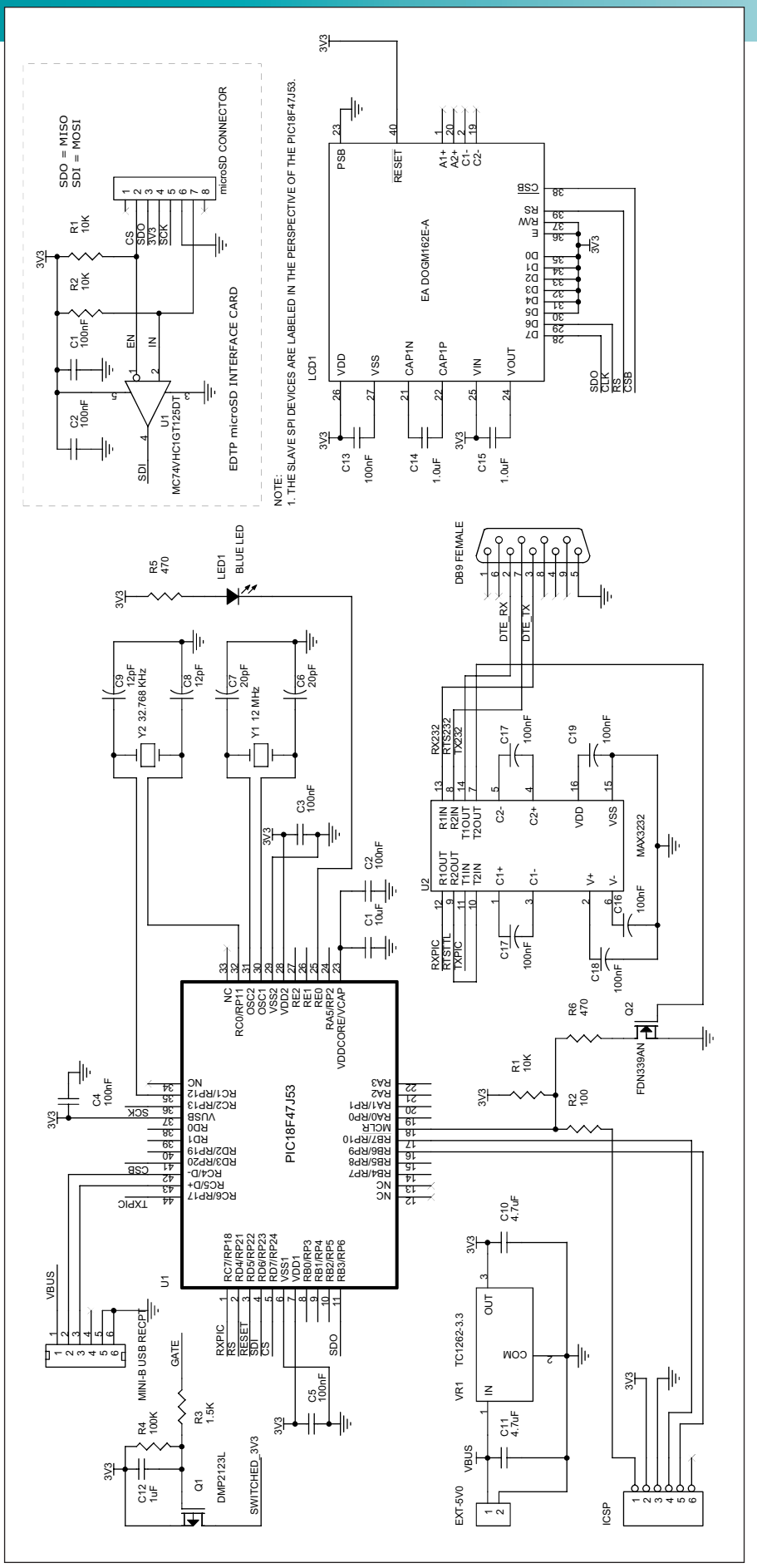
Proven Hardware

What better way to prove out our bootloader than to build it up on microcontroller hardware that has been previously proven to work. We'll use the EDTP Electronics USCM-47J53 eight-bit microcontroller board which is based on the Microchip PIC18F47J53. The USCM-47J53 is an eight-bit microcontroller platform that supports USB device functionality. The native PIC18F47J53 real time clock calendar subsystem is supported by an on-board 32.768 kHz crystal. If the USCM-47J53 in **Photo 1** looks familiar, that's because we put it through its paces in the December '10 and January '11 editions of *Nuts & Volts* Design Cycle. Thus far, the USCM-47J53 has hosted a USB HID application, a microSD card application, a USB CDC application, and acted as the base hardware for an SPI-based LCD project. The PIC18F47J53 has a couple of EUSARTs. We'll call upon the PIC's EUSART #1 for our serial bootloader party.

If you take a look back at our SPI-based LCD project, you'll see that we assigned the EUSART pins to duties other than acting as a serial port. If you don't have the January '11 issue of *Nuts & Volts* handy, you can grab a quick glimpse of the SPI-based LCD project schematic on the EDTP website. You might also be able to view the schematic via the back issues section of the *Nuts & Volts* website. The bottom line is that we need to make some changes to the original SPI-based LCD circuitry to free up the PIC's native EUSART1. Once the EUSART is exposed, we will fit it with a suitable RS-232 interface.

Our serial bootloader application revolves around the MAX3232 RS-232 transceiver. The nine-pin female D-shell connector and MAX3232 RS-232 transceiver are wired to form an RS-232 DCE (Data Communications Equipment) device. In simpler terms, the USCM-47J53 is wired to emulate a modem which means no special crossover cable is needed to connect to the PC's serial port. The PC's serial port is normally configured as a DTE (Data Terminal Equipment) device; that holds true in the case of my laptop's USB to RS-232 dongle. The primary difference in a DTE versus a DCE configuration lies in the wiring of the transmit (TX) and receive (RX) signal lines. Note in **Schematic 1** that the DTE RX signal is pinned directly to the DCE TX signal and the DTE TX signal is directly attached to the DCE RX signal. This pin-to-pin relationship between the DTE and DCE devices forms the data portion of a null modem. A full null modem implementation involves the RS-232 control signals which consist of DTR (Data Terminal Ready), DSR (Data Set Ready), RTS (Request to Send), and CTS (Clear to Send). For our purposes, the cross-tied transmit and receive lines will be enough to perform the data transfer job.

Since we are not utilizing the standard RS-232 DTE and DCE control signals, we can put the DTE-generated RTS control signal to work in another capacity. The RTS signal emanates from a DTE device and is intended to see if the



SCHEMATIC 1. The base USCM-47J53 board is truly universal as we've tied on a microSD interface, an SPI-based LCD, and a regulation RS-232 interface without any changes to the original design.

coast is clear to transmit. An RTS signal is normally acknowledged by the DCE device with the activation of its CTS signal. Obviously, there's no need for us to do that here.

We will use the RTS signal to drive the gate of Q2 — an FDN339AN MOSFET. When the FDN339AN is activated, the PIC's MCLR line is pulled logically low and the USCM-47J53 is reset. Thus, the FDN339AN is nothing more than an electronic RESET switch which allows the USCM-47J53 to be reset by the DTE device. We can also reset the USCM-47J53 using instructions embedded in the data stream. Once again, *Fred Eady's First Law of Embedded Computing* — which states that nothing is free — applies here. The FDN339AN reset method is not prone to data stream errors that may cause an unexpected reset. However, the FDN339AN reset method can only be implemented in a wired bootloader environment. Here's how the RTS reset operation looks from a C++ perspective:

```
void Comm::assertReset(void)
{
     serial->setRts(true);
     serial->setDtr(false);
}
void Comm::releaseReset(void)
{
     serial->setRts(false);
     serial->setDtr(true);
}
```

To utilize the RTS signal as a logic control signal, we must first convert the incoming RTS RS-232 voltage level to a CMOS logic level that is compatible with the +3.0 volt system used by the PIC18F47J53. Asserting the RTS signal at the DTE device causes a positive voltage to appear at the R2IN pin of the MAX3232. That translates to a logical low CMOS-compatible voltage at the MAX3232's R2OUT pin. Obviously, a logical high is needed to activate the FDN339AN. So, we push the logical low at R2OUT to the T2IN input. The result is a positive voltage at T2OUT. The output voltage at T2OUT is bi-polar and swings between -5.0 volts and +5.0 volts.

Although we have exposed the PIC's EUSART, we have not sacrificed one bit of

the USCM-47J53's original functionality. The USB and SPI components of the PIC will still operate normally. With that, let's get with the process of manually loading the bootloader core.

Bootloader Things You Need to Know

Every microcontroller has a memory subsystem, as well as rules that pertain to its particular memory layout. To guarantee the survival of our bootloader core code, we must assemble our bootloader application to adhere to the rules of the target microcontroller's memory map. In that we are working with the PIC18F47J53, it would be nice to get the skinny on how it utilizes its memory resources.

We added the MAX3232 RS-232 transceiver and supporting glue components to the base USCM-47J53 using caveman point-to-point wiring techniques. We won't be using rocks and sticks to assemble our bootloader core, however. Instead, we will take what we need from the Microchip Application Note AN1310. Here's some need-to-know PIC18F47J53 memory information taken from the AN1310 bootloader core code:

```
#ifdef __18F47J53
      #define CONFIG_AS_FLASH
      #define DEVICEID                  .711
      #define WRITE_FLASH_BLOCKSIZE     .64
      #define ERASE_FLASH_BLOCKSIZE     .1024
      #define END_FLASH                 0x20000
      #define END_GPR                   0xEB0
#endif
```

We are primarily interested in the blocksize and end boundary definitions. What the PIC device code snippet tells us is that it writes Flash memory in 64 byte blocks and erases Flash memory in 1,024 byte blocks. My HP-16C tells me that the 0x20000 hexadecimal END_FLASH definition equates to the PIC's 128K (131,072) of Flash. GPR in this instance is short for General Purpose Registers. The GPRs are located in the data memory area. The PIC datasheet states that the total SRAM area contained within it is 3.8 Kbytes. The END_GPR definition tells us that 3,760 bytes of SRAM exists within the PIC.

Just for grins — using the decimal value of 711 as a seed — I researched the DEVICEID value and came up with this binary sequence:

```
0b01011000111
```

If you group the bits as to read 0x02C7, the bit pattern identifies the Device ID of the PIC18F47J53. This is the bit pattern that is written into the Device ID bits of the PIC. The Device ID is used to identify the microcontroller to hardware programmers and applications. However, we won't concern ourselves with the Device ID unless we have to. Now that we have the PIC's memory extent information, let's learn more about the bootloader core firmware.

Our bootloader core firmware is written in assembler and is supported by the *preprocess.inc*, *devices.inc*, and *bootconfig.inc* files. The PIC18F47J53 memory extents we examined earlier were pulled from the *devices.inc* file.

We already know that we will place our bootloader core firmware at the end of program Flash. The code and definitions within the *preprocess.inc* determines exactly where the bootloader core will reside in high program Flash. We will need the *devices.inc* memory extent information and some code size information from the *bootconfig.inc* file to assist in pinpointing the bootloader core firmware's absolute address in program Flash:

```
#define BOOTLOADERSIZE  .708
;From bootconfig.inc

;From preprocess.inc
#if BOOTLOADERSIZE < ERASE_FLASH_BLOCKSIZE
     ; This device has a large Erase FLASH Block
     ; Size, so we need to reserve a full Erase
     ; Block page for the bootloader. Reserving
     ; an entire erase block prevents the PC
     ; application from trying to accidently
     ; erase a portion of the bootloader.
     #define BOOTBLOCKSIZE ERASE_FLASH_BLOCKSIZE
     #ifndef BOOTLOADER_ADDRESS
          #ifdef CONFIG_AS_FLASH
               #define BOOTLOADER_ADDRESS
               (END_FLASH - BOOTBLOCKSIZE -
               ERASE_FLASH_BLOCKSIZE)
          #else
               #define BOOTLOADER_ADDRESS
              (END_FLASH - BOOTBLOCKSIZE)
          #endif
     #endif
```

I have culled only the *preprocess.inc* code that pertains to our USCM-47J53. Our BOOTLOADERSIZE of 708 bytes is less than the ERASE_FLASH_BLOCKSIZE of 1,024 bytes. If you flow through the *preprocess.inc* code, you'll end up calculating the BOOTLOADER_ADDRESS as:

```
0x20000 – 0x00400 – 0x00400 = 0x1F800
```

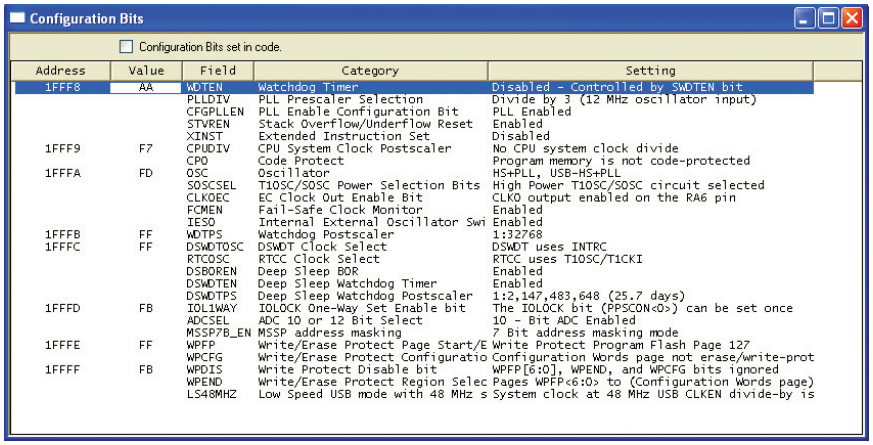
There is no need to modify any of the bootloader core files we've mentioned. All we need to do is make sure that the PIC configuration bits are set up correctly. When we program the bootloader core firmware into the PIC, its configuration bits will also be programmed. This means that our downloaded application must operate with the same configuration bits that are used by the bootloader. It is not recommended to change the configuration bits with each application download as a mixup in the configuration bits could cause the USCM-47J53 to fail to execute the newly downloaded application. The bootloader core firmware

Sources

Microchip
Application Note AN1310
PIC18F47J53
PICkit3
www.microchip.com

EDTP Electronics
USCM-47J53
www.edtp.com

Fred Eady may be reached via email at fred@edtp.com.



SCREENSHOT 1. This is a capture of the MPLAB Configuration Bits window. Basically, we have enabled the PLL to clock at 48 MHz, turned off the WDT, and enabled the RTCC.

configuration bits are outlined in **Screenshot 1**.

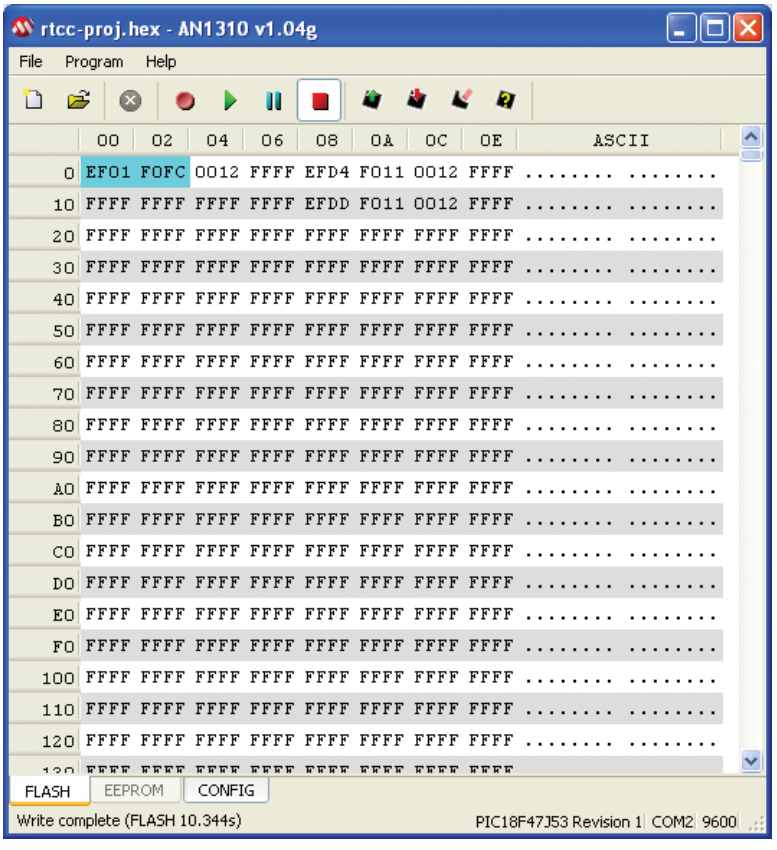
Before we actually load the bootloader core firmware, let's assemble the bootloader core source and check the resultant map file for the bootloader start point and size:

```
 Program Memory Usage
      Start      End
      -----      -----
    0x01f800   0x01fa5f
608 out of 131074 program addresses used,
program memory utilization is 0%
```

The excerpt from the *PIC18 Bootloader.map* file indicates that our bootloader core is 608 bytes in length and begins at 0x01F800. We're ready to load it up!

Loading the Gun

I used a PICkit3 to manually program the bootloader core hex file into the PIC18F47J53. If you consider the bootloader a "gun," then the "bullets" are the applications the gun shoots down to the PIC. The gun is loaded using the PC bootloader application included in the AN1310 code package. In **Screenshot 2**, I've taken the liberty to load the gun with the PIC18F47J53 RTCC/LCD application we built in the January '11 Design Cycle. As you can see, I fired the bullet at 9600 bps. The PC bootloader control program can push bits down the wire in excess of 1 Mbps. The maximum transfer speed depends on the capabilities of the target's UART. The USCM-47J53 runs with a 48 MHz system clock which limits the PC bootloader control application's maximum transfer rate to 921600 bps. The bootloader control program running on the PC communicates with an autobaud scheme coded into the bootloader core firmware. The autobaud algorithm will not allow you to connect with a baud rate that is too high for the PIC18F47J53 to handle.



SCREENSHOT 2. Application Note AN1310 also comes with a PC application that communicates with the bootloader core firmware. As you can see in this capture, I'm ready to load the RTCC application via the bootloader.

Test Firing the Gun

The PIC18F47J53 RTCC/LCD application I shot at the USCM-47J53 with the bootloader gun ran just as if the bootloader core firmware was not there. And better yet, the cost of our serial bootloader project was minimal. If your microcontroller target already has a regulation RS-232 port, your cost to implement the bootloader we just discussed is the cost of an FDN339AN MOSFET.

Why Not USB?

As you are well aware, bootloading can be accomplished via USB just as easily as we invoked our RS-232 bootloader. However, not every PIC has native USB capability. On the other hand, most PICs have native RS-232 capability. There aren't many times you can say that an RS-232 implementation beat out a similar USB design. Not only was our RS-232 monetarily inexpensive, we implemented our RS-232 bootloader without having to write a single byte of code! SV

Build Your Own Big Walker Part 3

by Daniel Albert

www.servomagazine.com/index.php?/magazine/article/april2011_Albert

This segment focuses on the design and implementation of the big walker's multi-processor distributed processing system. This part in the series is crucial in simplifying the algorithm that creates a simple walk. Rather than one processor controlling all the servos and reading all of the sensors, these tasks are split into individual processors. A logical place to divide the system is at the limb level. The same tasks required to control the left leg can control the right leg. Each limb therefore has its own processing board (Limb Processing Board) or LPB.

Distributed Processing

The LPB is designated with the tasks of handling that limb's sensor input, servo control, and communications with particular other boards. Most of today's MCUs can easily handle this. I used a Microchip dsPIC30F4012. It can run at 30 MIPs. For now, the big walker has only two limbs and two LPBs. However, in time, two arms will be added.

The more complex a system is, the harder it is to debug. The distributed system simplifies the debugging task by applying a simple debugging tactic. That is: "It is not so much as **what** the problem is, as to **where** the problem is." By isolating tasks to separate processors, we can simplify the "where." As with most projects, they evolve in what is termed "feature creep." We just can't help adding more functions. This is where a single board computer system becomes harder and harder to debug. Like the feather that broke the camel's back, many times one extra line of code in a complex system can allow subtle time or resource related bugs to develop in the code in an unrelated area. A distributed system can be naturally immune to this. A fixed set of high priority tasks that are isolated within one MCU can never be hogged by some other unrelated task assigned to another MCU. That cannot be guaranteed with a single processor with a single or multi-threaded operating system.

Micro Controller Units (MCUs) unlike Central Processing Units (CPUs) are designed to control some external hardware. Typically, there are time constraints required to handle hardware inputs and outputs. Interrupt Service Requests (ISR) are the code segments — or handlers — that process these signals. They have top priority of the MCU. This is confusing to some people, but ISRs are the foreground tasks in an MCU. There are priority levels to each ISR and the highest priority is the task most in the foreground. Main() code is the lowest priority or background task which is only run when an ISR is not running. It is sometimes easier to think of this as urgency vs. importance. Foreground tasks are urgent. Sensor reading and servo driving of each independent limb are time critical. They MUST be done on time. Missing a sensor input or improperly driving a servo is not an option. Background tasks tend to be important but not as time critical. Making the right decision from the sensor input is very important. Passing information to other parts of the system is also very important. For some, it may be easier to think of urgency and importance in time. In the case of the big walker, urgent tasks tend to be handled in

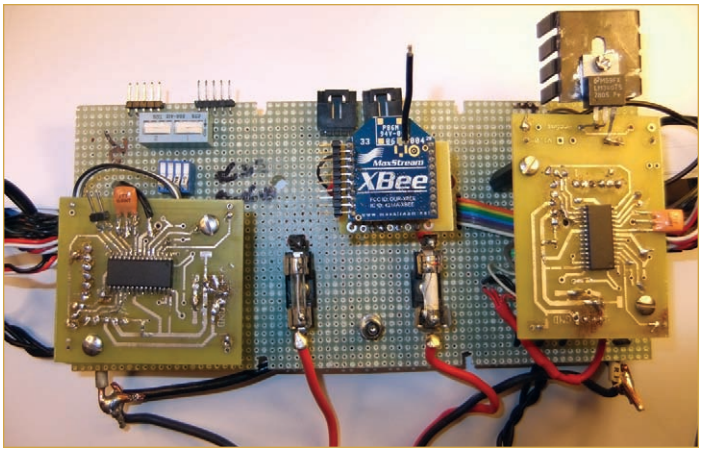


FIGURE 1. Notice the two Limb Processing Boards (one for each leg). The ZigBee adapter is in the middle. The board everything is mounted on is the power and I²C bus board.

TABLE 1. Limb Processing Board (LPB) high priority interrupt driven foreground tasks. These are worst-case 100% throughput scenarios for an LPB master. Actual time is considerably less.

Task	Frequency	Process Time	Total MCU Time
Load Cell 1	9,091 Hz	5 µSec	45.5 mSec
Load Cell 2	9,091 Hz	5 µSec	45.5 mSec
Load Cell 3	9,091 Hz	5 µSec	45.5 mSec
Eight Servos (Bit-Banged)	50 Hz	5 µSec times 8	2 mSec
I²C Packet TX/RX	1,920 Hz	30 µSec	58 mSec
Serial Port TX/RX 9600 baud	1,920 Hz	8 µSec	15 mSec
Total MCU Time			211.5 mSec (21.15%)

TABLE 2. Limb Processing Board (LPB) background tasks. These are worst-case 100% throughput scenarios for an LPB master. Actual time is considerably less.

Task	Frequency	Process Time	Total MCU Time
Load Cell 1 Calculation	600 Hz	80 µSec	48 mSec
Load Cell 2 Calculation	600 Hz	80 µSec	48 mSec
Load Cell 3 Calculation	600 Hz	80 µSec	48 mSec
Eight Servos Setup	50 Hz	40 µSec	2 mSec
I²C packet TX/RX Setup	1,920 Hz	10 µSec	19.2 mSec
Serial Port Pass-through 9600 baud	1,920 Hz	10 µSec	19.2 mSec
I²C or Serial Command Interpreter	4,800 Hz	45 µSec	216 mSec
Auto Mode CG Routine	100 Hz	80 µSec	8 mSec
Total			408.4 mSec (40.84%)

microseconds (millionths) and important tasks need to be handled in milliseconds (thousandths).

To guarantee that all high priority tasks can be handled, we will profile the code and create a table of their time to completion. This is usually done with the help of a logic analyzer. Pick some general-purpose I/O port and flip the bit to high when entering the ISR or task, and low upon leaving. The logic analyzer can then be used to indicate the time spent in that ISR. If you do not have a logic analyzer, get one!

Table 1 and **Table 2** show the list of tasks for ISRs and background tasks.

The big walker depends heavily (pun intended) on the load cells to supply fast and accurate weights. Each load cell must be read separately. The load cells used are capacitive vs. resistive. Instead of an analog voltage, they output a digital frequency between 100-150 kHz. This requires the edge capture port of the MCU to determine the exact frequency. The capture port can generate and interrupt on one, four, or 16 edge counts. Even at 16 edges (or about once every 106 µSec), this is a frequency of 9,375 per second (150,000/16). Rather than try to calculate a weight at this rate, the driver waits for 16 interrupts. That slows down the conversion rate to (9,375 Hz/16 = 586 Hz) or 1.7 mSecs. The driver saves an internal register that contains the MCU timer tics since the last interrupt to calculate the weight. The actual calculation of weight is non-trivial for capacitive sensors. Unlike a resistive sensor's voltage output, the frequency output calculation to weight is non-linear. A set of quadratic parameters comes with each sensor. This calculation is performed in the background (not in a driver interrupt) since it only happens at 586 Hz.

The servos use the standard R/C (Radio Controlled) PWM (Pulse Width Modulation) style

TABLE 3. Simple Two-Letter Commands.

Command	Parameter	Description	Returns	Packet Length
Tn	n = 0,1, 2 or A(all)	Tares (sets to zero) the load cell.	A	TX = 2; RX = 1
Wn	n = 0,1, 2 or A(all)	Returns single weight or all three (comma delimited).	99.99,99.99, 99.99,99.99	TX = 2; RX = 5,17
Cx	x = M or S	Configure as a master or slave.	A	TX = 2; RX = 1
Cx	x = R or L	Configure right or left footedness.	A	TX = 2; RX = 1
Sn pppp	n = servo (0 - 5) p = position (500 - 2500)	Set a servo to a fixed position. 500 = -90 degrees; 2,500 = +90 degrees	A	TX = 2; RX = 1
Pn	n = servo (0 - 5) p = position (500 - 2500)	Get a servo's position. 500 = -90 degrees; 2,500 = +90 degrees	9999	TX = 2; RX = 4
Mx	x = A or M	Balancing mode set to auto or manual.	A	TX = 2; RX = 1
SP	none	Save current servo values as initial position.	A	TX = 2; RX = 1
IP	none	Set all servos to their initial position.	A	TX = 2; RX = 1
SA n.nnnnEsxx	n.nnnn = float s= sign xx = exponent	Set load cell quadratic parameter A.	A	TX = 13; RX = 1
SB n.nnnnEsxx	n.nnnn = float s= sign xx = exponent	Set load cell quadratic parameter B.	A	TX = 13; RX = 1
SC n.nnnnEsxx	n.nnnn = float s= sign xx = exponent	Set load cell quadratic parameter C.	A	TX = 13; RX = 1
GA		Get load cell quadratic parameter A.	n.nnnnEsxx	TX = 2; RX = 10
GB		Get load cell quadratic parameter B.	n.nnnnEsxx	TX = 2; RX = 10
GC		Get load cell quadratic parameter C.	n.nnnnEsxx	TX = 2; RX = 10

drive signal. Briefly, the signal has a frequency of 50 Hz (20 mSecs). Within the 20 mSecs is a duty cycle (on period) of anywhere from .5-2.5 mSecs. The remaining time of 19.5-17.5 mSecs is deadband (off period). The on period corresponds to degrees. The middle travel point is at zero degrees or 1.5 mSecs. The servo has about 200 degrees of movement (-100 to +100) and a resolution of .1 degree. Each degree is 100 µSec of the duty cycle. For 1/10 of one degree of resolution, a 10 µSec edge discrimination is needed. The LPB bit-bangs (manually drives) these signals. It takes less than 5 µSec to handle the timer interrupt that drives the servo. Even though there are only five servos per limb in this design, the program can easily handle up to eight per limb.

The bulk of the MCU's remaining time is spent communicating. While the I²C port is interrupt driven, the data that is received and sent is performed by the background task. At 400 kHz, the I²C bus can send a 10-byte packet in about 250 µSec. However, the commands originate from the PC at a rate of 9600 baud full duplex. Therefore, the I²C bus will never be saturated. It effectively runs at the total serial port throughput of 19200 baud; an addition of 9600 baud TX plus 9600 baud RX. A slave may not even respond at all, so a 20 mSec time-out is also allotted per slave.

The maximum total MCU usage of 21.15% for foreground tasks and 40.84% for background tasks are extremely conservative numbers. In actual testing, the numbers are less than half. So, one might get the impression that this whole process for all limbs could be done with one larger 200 MIPs processor skipping the I²C communications completely. True, but then as stated above, we would lose the benefits of the distributed processing design. Another reason the LPB's MCU has more processing power than seems necessary is that one of the LPBs will be designated as a "master" while all the others are "slaves." A master has the added job of communicating wirelessly with an external processor that is running the walking algorithm. It bridges those wireless commands to the I²C bus. This design needs only one I²C master (all other nodes are slaves). Since the I²C communications are mostly a background task, the master's standard foreground tasks do not suffer any latency.

This is prototype firmware and is much easier to debug if there is a text driven command interface. Each LPB has an I²C command interface which responds to short, simple two-letter text commands. By keeping the commands short, the command interface can be manually controlled from a standard terminal program (like Hyperterm) or automatically controlled by a custom program or script. The ZigBee enabled LPB master receives these wireless commands from a PC also equipped with a ZigBee-to-serial wireless model. If the commands are for an LPB slave, the master will forward them to the appropriate MCU via the I²C bus. The slaves response to the master is then transmitted back to the PC. Parameters can be set, sensors read, and servos positioned anywhere in the robot's distributed network. Commands that do not return data will return 'A' as an acknowledgement. **Table 3** shows the current command set.

I²C Packet Format

I²C packets have the destination address built into the protocol. Each slave has three jumpers on the PC board to create up to eight distinct addresses. The length of each packet is predefined. The packets have a simple checksum — not counted in the length — that is added to the last byte. Commands received with a checksum that does not agree will return an E rather than data or an A.

A sample I²C command that tares sensor 0 (see **Table 3**) from a master looks like this. The master sends the data in bold; the slave replies with the non-bold data.

address	**data**	**data**	**chksum**
0x52	**0x54(T)**	**0x30(0)**	**0xD6**

(0x52 + 0x54 + 0x30)

address	data	chksum
0x52	0x65(A)	0xB7 (0x52 + 0x65)

There are some configurations that need to be saved. Settings such as initial servo positions, left and right footedness, load cell parameters, and master/slave are stored in the EEPROM automatically when changed. Upon reboot, these are read from EEPROM.

Next month, I will pull together the first three parts of this series and describe how the walking algorithm uses these defined commands for a slow but simple walk without gyros or accelerometers. **SV**

6th ANNUAL

Maker Faire®

THE World's Largest DIY Festival

A two-day, family-friendly event to MAKE, create, learn, invent, CRAFT, recycle, think, play, celebrate, and be inspired by arts, crafts, engineering, food, music, science, and technology.

COME PLAY WITH 600+ MAKERS OF ALL AGES!

SAVE THE DATES!

Maker Faire Bay Area
May 21 & 22, 2011
San Mateo County
Event Center
San Mateo, CA

Maker Faire Detroit
July 30 & 31, 2011
The Henry Ford
Dearborn, MI

World Maker Faire
September 17 & 18, 2011
New York Hall of Science
Flushing Meadows
Corona Park, Queens, NY

To participate and get more info, **check out the website »**

MakerFaire.com

Brought to you by MAKE magazine

The SERVO Webstore

Attention Subscribers ask about your discount on prices marked with an *

CD-ROM SPECIALS

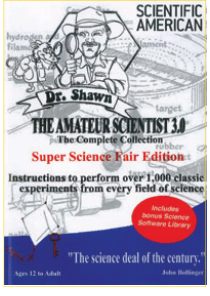
ROBOTICS

PIC Robotics
by John Iovine

Here's everything the robotics hobbyist needs to harness the power of the PICMicro MCU!

In this heavily-illustrated resource, author John Iovine provides plans and complete parts lists for 11 easy-to-build robots each with a PICMicro "brain." The expertly written coverage of the PIC Basic Computer makes programming a snap – and lots of fun.
$24.95

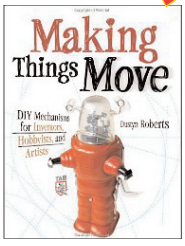
The Amateur Scientist 4.0 The Complete Collection
by Bright Science, LLC



There are 1,000 projects on this **CD**, not to mention the additional technical info and bonus features. It doesn't matter if you're a complete novice looking to do your first science fair project or a super tech-head gadget freak; there are enough projects on the single CD-ROM to keep you and 50 of your friends busy for a lifetime!
Reg $26.95 Sale Price $23.95

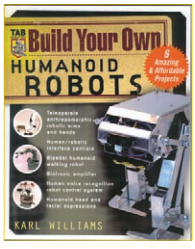
Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists
by Dustyn Roberts

NEW!



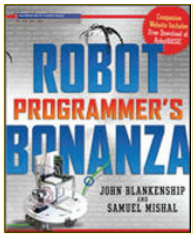
In *Making Things Move: DIY Mechanisms for Inventors, Hobbyists, and Artists*, you'll learn how to successfully build moving mechanisms through non-technical explanations, examples, and do-it-yourself projects — from kinetic art installations to creative toys to energy-harvesting devices. Photographs, illustrations, screenshots, and images of 3D models are included for each project.
$29.95*

Build Your Own Humanoid Robots
by Karl Williams
GREAT 'DROIDS, INDEED!



This unique guide to sophisticated robotics projects brings humanoid robot construction home to the hobbyist. Written by a well-known figure in the robotics community, *Build Your Own Humanoid Robots* provides step-by-step directions for six exciting projects, each costing less than $300. Together, they form the essential ingredients for making your own humanoid robot.
$24.95*

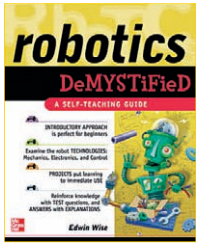
Robot Programmer's Bonanza
by John Blankenship, Samuel Mishal



The first hands-on programming guide for today's robot hobbyist!
Get ready to reach into your programming toolbox and control a robot like never before! *Robot Programmer's Bonanza* is the one-stop guide for everyone from robot novices to advanced hobbyists who are ready to go beyond just building robots and start programming them to perform useful tasks.
$29.95

Robotics Demystified
by Edwin Wise

YOU DON'T NEED ARTIFICIAL INTELLIGENCE TO LEARN ROBOTICS!



Now anyone with an interest in robotics can gain a deeper understanding — without formal training, unlimited time, or a genius IQ. In *Robotics Demystified*, expert robot builder and author Edwin Wise provides an effective and totally painless way to learn about the technologies used to build robots! **$19.95**

We accept VISA, MC, AMEX, and DISCOVER
Prices do not include shipping and may be subject to change.

To order call 1-800-783-4624

SERVO Magazine Bundles

Published by T & L Publications, Inc.

$57 per bundle

Save $10 off the normal price!!

Now you can get one year's worth of all your favorite articles from *SERVO Magazine* in a convenient bundle of print copies. Available for years 04, 05, 06, 07, 08, and 09.

Call my Webstore and you'll get someone in AMERICA!

Visit my online store @ www.servomagazine.com

LEGO MINDSTORMS NXT Idea Book

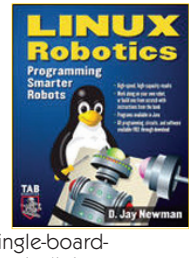
by the Contributors to The NXT Step Blog

If you're serious about having fun with LEGO® robotics, you've come to the right place. The team behind The NXT STEP blog – the authoritative online source for MINDSTORMS® NXT information and advice – has packaged its considerable skills and experience in this book. Inside, you'll find some of the team's best ideas for creating cool and sophisticated models, including instructions for eight robots you can build yourself.

Reg $29.95 Sale Price $24.95

Linux Robotics

by D. Jay Newman

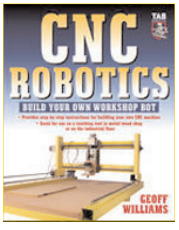


If you want your robot to have more brains than microcontrollers can deliver – if you want a truly intelligent, high-capability robot – everything you need is right here. *Linux Robotics* gives you step-by-step directions for "Zeppo," a super-smart, single-board-powered robot that can be built by any hobbyist. You also get complete instructions for incorporating Linux single boards into your own unique robotic designs. No programming experience is required. This book includes access to all the downloadable programs you need.

$34.95

CNC Robotics

by Geoff Williams

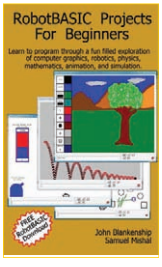


Here's the FIRST book to offer step-by-step guidelines that walk the reader through the entire process of building a CNC (Computer Numerical Control) machine from start to finish. Using inexpensive, off-the-shelf parts, readers can build CNC machines with true industrial shop applications such as machining, routing, and cutting – at a fraction of what it would cost to purchase one. Great for anyone who wants to automate a task in their home shop or small business. **$34.95**

SPECIAL OFFERS

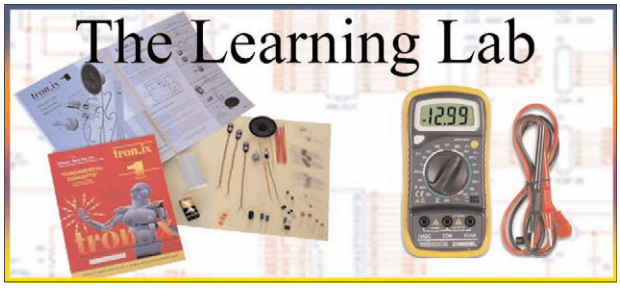
RobotBASIC Projects For Beginners

by John Blankenship, Samuel Mishal



If you want to learn how to program, this is the book for you. Most texts on programming offer dry, boring examples that are difficult to follow. In this book, a wide variety of interesting and relevant subjects are explored using a problem-solving methodology that develops logical thinking skills while making learning fun. RobotBASIC is an easy-to-use computer language available for any Windows-based PC and is used throughout the text.

Reg. Price $14.95 Sale Price $9.95



Technology Education Package for Everyone Starting in Electronics

This lab – from the good people at GSS Tech Ed – will show you 40 of the most simple and interesting experiments and lessons you have ever seen on a solderless circuit board. As you do each experiment, you learn how basic components work in a circuit. Along with the purchase of the lab, you will receive a special password to access the fantastic online interactive software to help you fully understand all the electronic principles. For a complete product description and sample software, please visit our webstore.

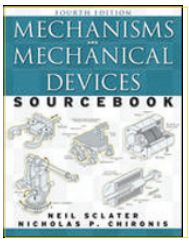
Regular Price $79.95 **Subscriber's Price $75.95**

Or order online
www.servomagazine.com

SPECIAL OFFERS

Mechanisms and Mechanical Devices Sourcebook

by Neil Sclater, Nicholas Chironis



Over 2,000 drawings make this sourcebook a gold mine of information for learning and innovating in mechanical design. Overviews of robotics, rapid prototyping, MEMS, and nanotechnology will get you up to speed on these cutting-edge technologies. Easy-to-read tutorial chapters on the basics of mechanisms and motion control will introduce those subjects to you. **Reg $89.95 Sale Price $69.95**

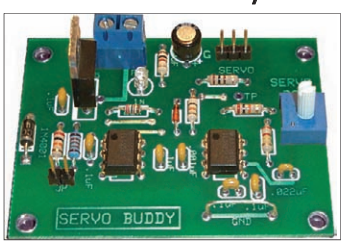
Forbidden LEGO

by Ulrik Pilegaard / Mike Dooley

Forbidden LEGO introduces you to the type of free-style building that LEGO's master builders do for fun in the back room. Using LEGO bricks in combination with common household materials (from rubber bands and glue to plastic spoons and ping-pong balls) along with some very unorthodox building techniques, you'll learn to create working models that LEGO would never endorse. **Reg $24.95 Sale Price $19.95**

PROJECTS

The SERVO Buddy Kit

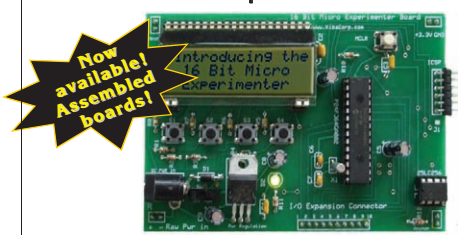


An inexpensive circuit you can build to control a servo without a microcontroller.

For more information, please check out the May 2008 issue or go to the SERVO webstore.

Includes an article reprint.
Subscriber's Price **$39.55**
Non-Subscriber's Price **$43.95**

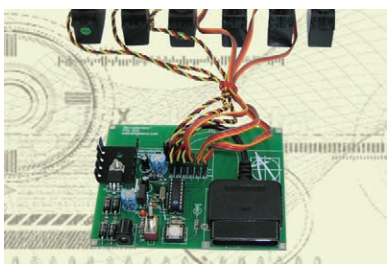
16-Bit Micro Experimenter Board



Ready to move on from eight-bit to 16-bit microcontrollers? Well, you're in luck! In the December 2009 *Nuts & Volts* issue, you're introduced to the 16-Bit Micro Experimenter. The kit comes with a CD-ROM that contains details on assembly, operation, as well as an assortment of ready-made applications. New applications will be added in upcoming months.

Subscriber's Price **$55.95**
Non-Subscriber's Price **$59.95**

PS2 Servomotor Controller Kit



This kit accompanied with your own PlayStation controller will allow you to control up to six servomotors. Includes all components and instruction manual.

For more information, please see the February 2011 edition of *SERVO Magazine*. Assembled units available!

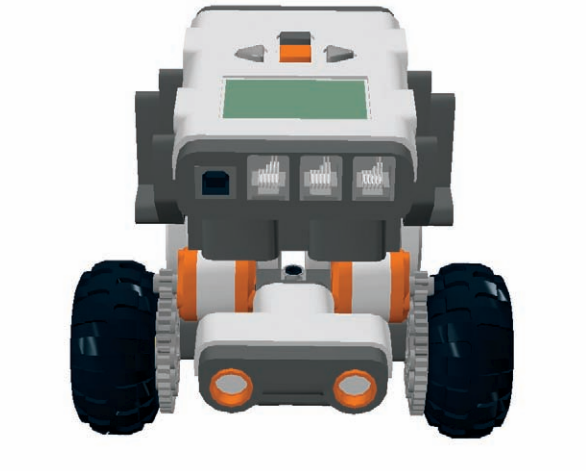
Subscriber's Price
$79.95
Non-Subscriber's Price
$84.95

The NXT Big Thing #9

Rad Radar!

By Greg Intermaggio

Hey all! In last month's column, we introduced the compass sensor — an awesome addition to the variety of sensors available for the NXT. We used the compass sensor to keep Eddie heading in a specific direction, and programmed him such that even if you turn him around, he will turn back to the correct direction and continue on.



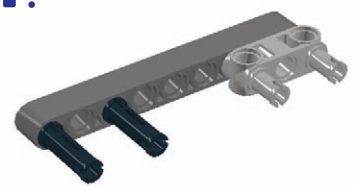
This time, we'll add an ultrasonic sensor and write a new program for Eddie so that he can head in a specific direction while avoiding obstacles. We'll make it so that when Eddie is heading north and about to hit a wall, he turns to get around it.

This article again assumes that you have the NXT compass sensor, available from **HiTechnic.com** – go check out their awesome array of sweet sensors for LEGO MINDSTORMS.

Now, let's get building!

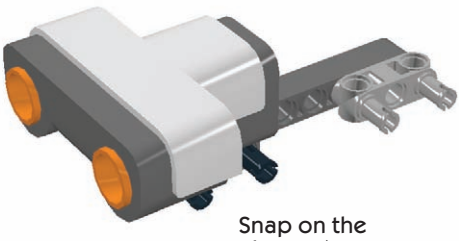
Building Instructions: Ultrasonic Sensor Attachment

1.



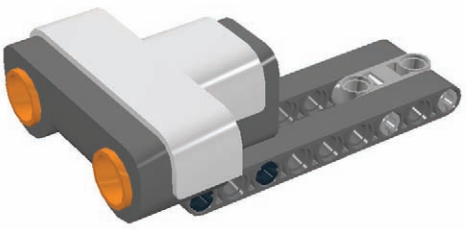
Start with a nine-hole studless beam with the indicated pieces installed.

2.

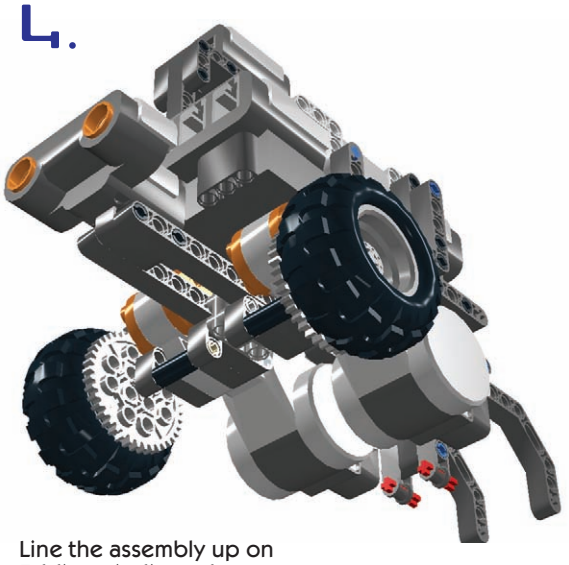


Snap on the ultrasonic sensor.

3. Add a second nine-hole studless beam to close off the other side.

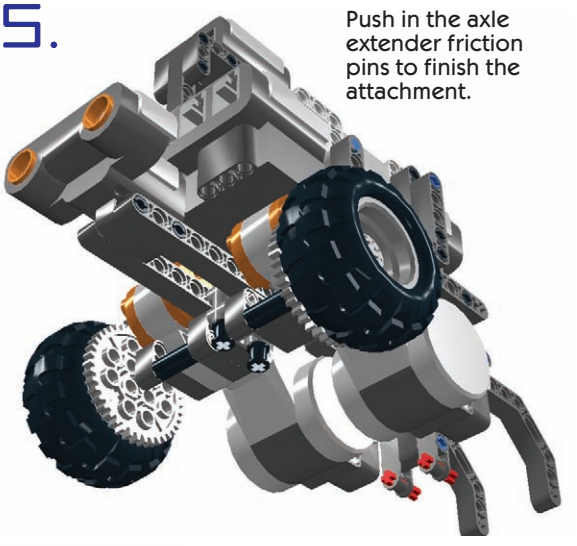


4.



Line the assembly up on Eddie as indicated.

5. Push in the axle extender friction pins to finish the attachment.



6.

Eddie with the compass sensor attachment from last month and the ultrasonic sensor attachment in all his glory!

Writing The Program

Now that we've got the ultrasonic attachment built, let's get to the program.

IMPORTANT NOTE: If you haven't already, you need to complete the main program from The NXT Big Thing #8 — Stay On Course!

If all's well and done, Eddie should now navigate to magnetic north while avoiding obstacles.

Program 1 Instructions

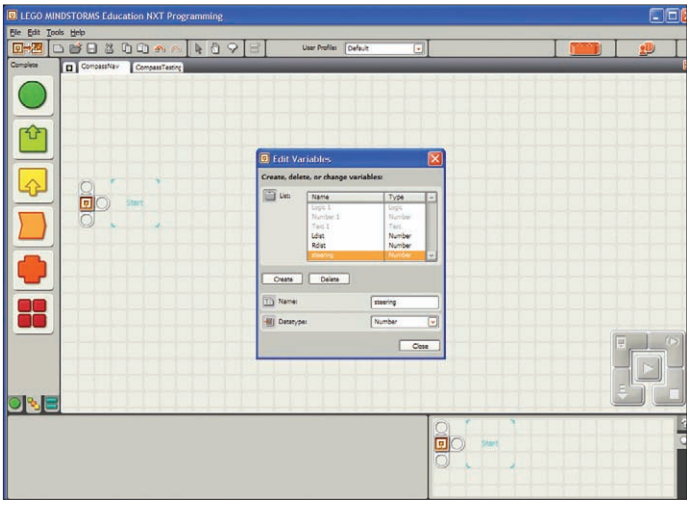


Figure 1. Create a new program called CompassNav, and open our old CompassTest program from last month. In CompassNav, click Edit > Define Variables and define Ldist, Rdist, and steering as number variables.

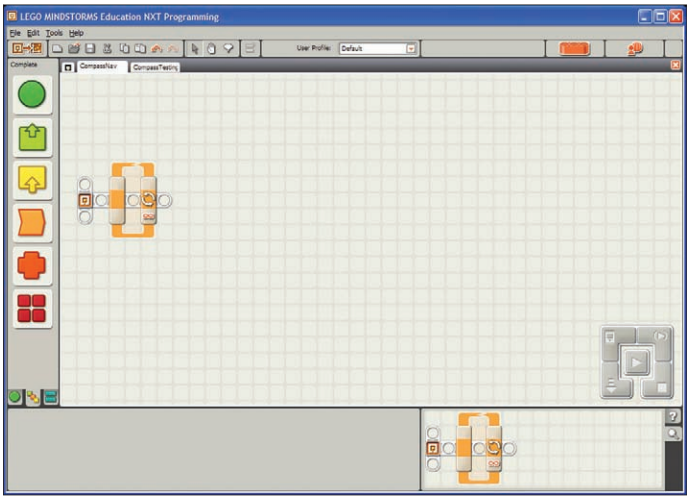


Figure 2. Then, get started by adding a loop.

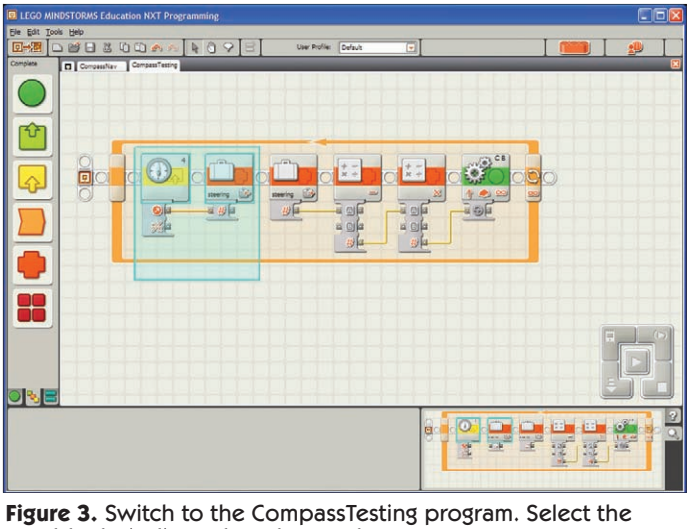


Figure 3. Switch to the CompassTesting program. Select the two blocks indicated, and copy them.

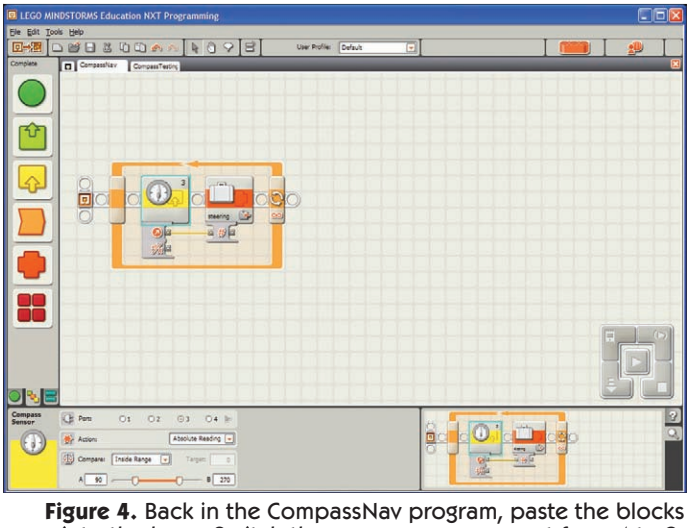


Figure 4. Back in the CompassNav program, paste the blocks into the loop. Switch the compass sensor port from 4 to 3.

More Challenges!

Ready for more? Here's a few things to try:

- Play around with the distance that Eddie reacts to obstacles, and the amount of time he spends trying to get around them. Try to find optimal settings.
- Make Eddie move south instead of north. Try making him move east or west, as well.
- Add another step to back away from an obstacle if

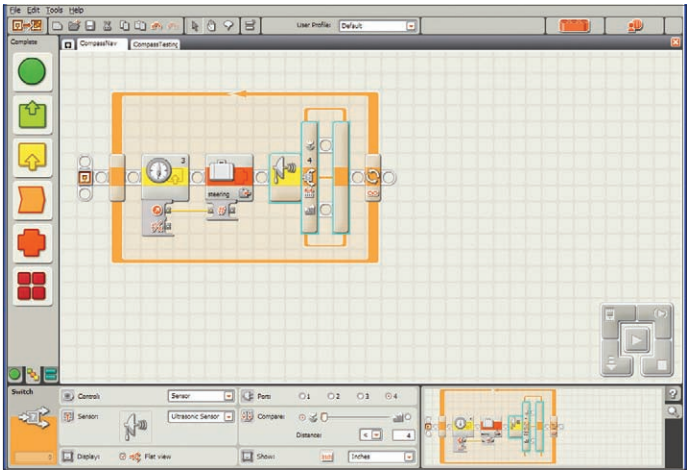


Figure 5. Under the Flow tab in the complete pallette, add a switch. Set the sensor to ultrasonic, the port to 4, and the distance to less than four inches. This will be the distance from a wall at which Eddie will decide how he wants to avoid that wall. You may want to play with this value later to see what's best for your room.

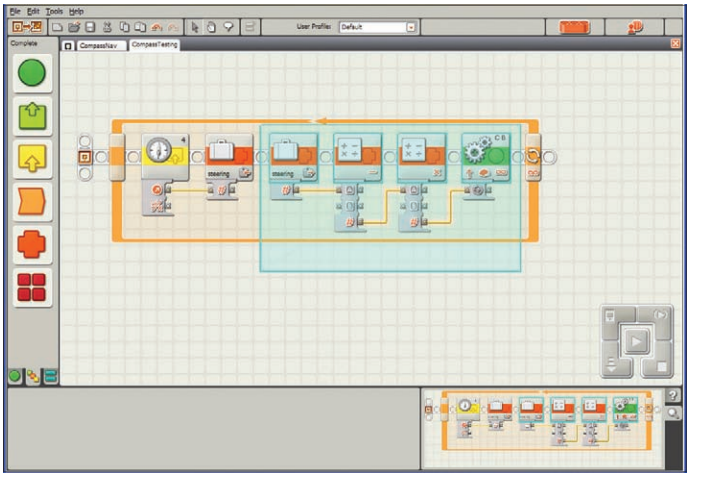


Figure 6. Hop back over to the CompassTest program. Select and copy the indicated blocks.

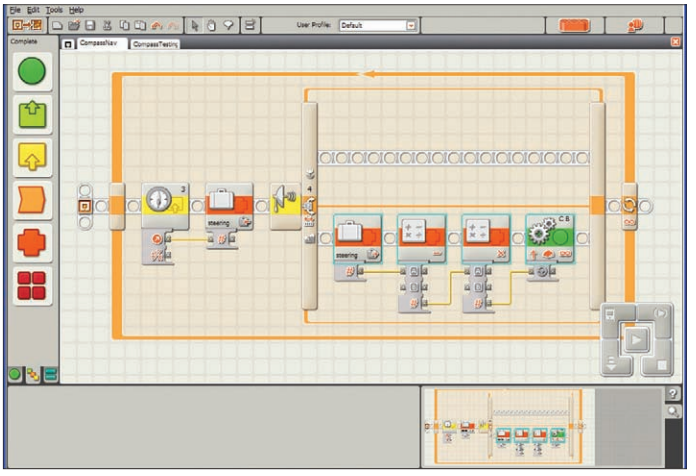


Figure 7. Paste the blocks into the ultrasonic switch, on the side with the mountains icon. This means that Eddie will perform our old CompassTest program when he is more than four inches away from an obstacle.

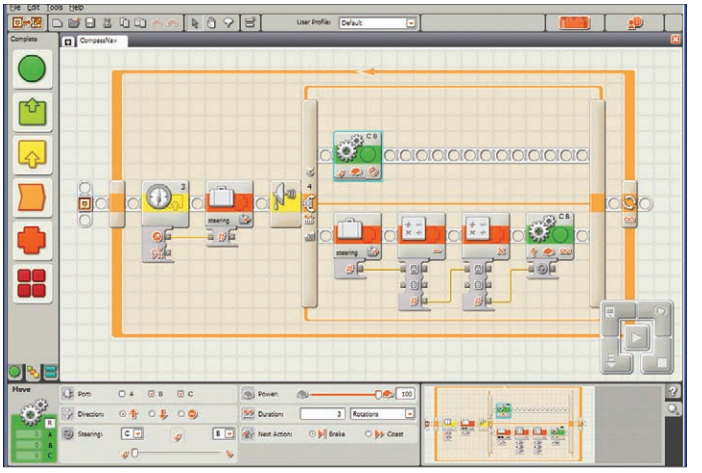


Figure 8. When Eddie is less than four inches from an obstacle, the first thing we want him to do is turn left to see if there's a clear path. Add a motor block turning left for three rotations at full power (you should be using an 8:40/1:5 gear ratio for Eddie for this exercise).

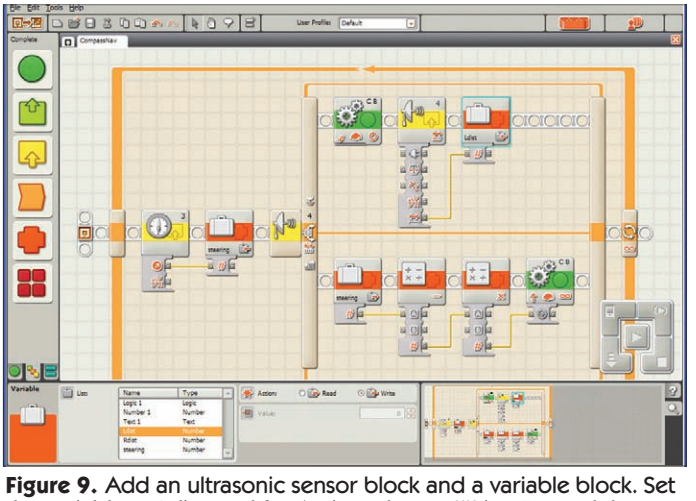


Figure 9. Add an ultrasonic sensor block and a variable block. Set the variable to Ldist, and for Action, choose Write. Expand the ultrasonic sensor block and run a data wire from Distance to Value.

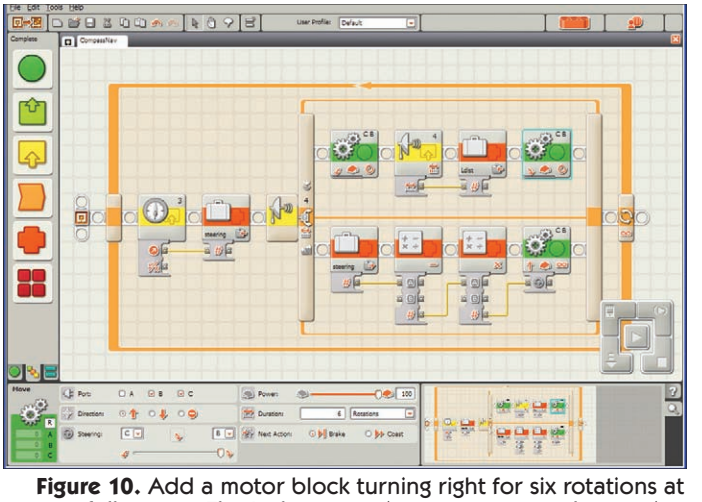


Figure 10. Add a motor block turning right for six rotations at full power. That's three rotations to return to the starting position, then three more to turn an equal distance to the right.

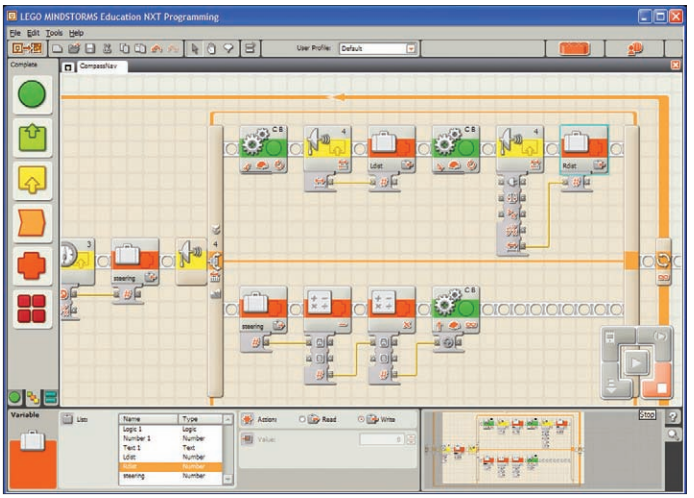


Figure 11. Add an ultrasonic sensor block and a variable block, and repeat step 9, except set the variable to Rdist. This will record how much clear space Eddie has to his right. Next up, we'll compare the left distance to the right distance, and have Eddie decide which way to go based on which one is clearer.

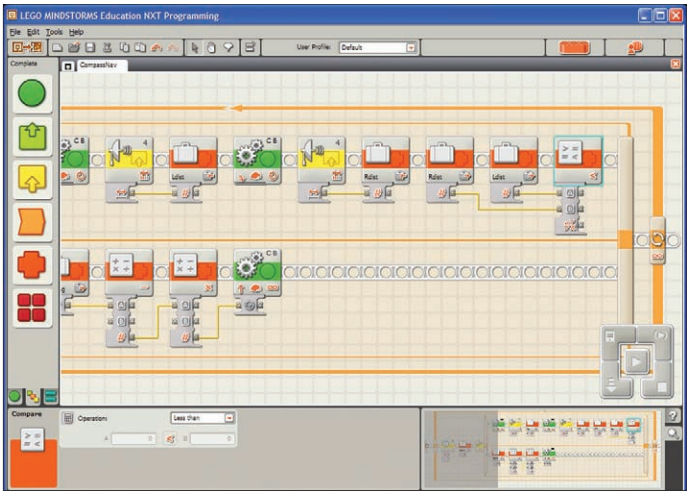


Figure 13. From the Data tab, add a compare block and expand its data hubs. Attach Ldist to A and Rdist to B, and make sure the operation is set to Less than.

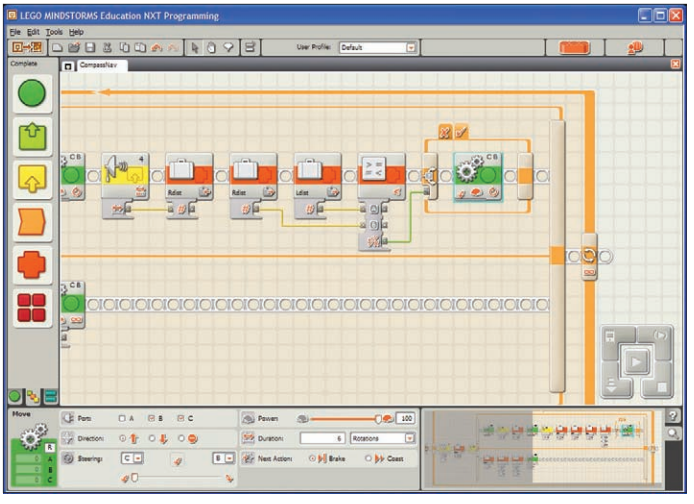


Figure 15. Click the other (True) tab on the switch. Add a move block turning left for six rotations.

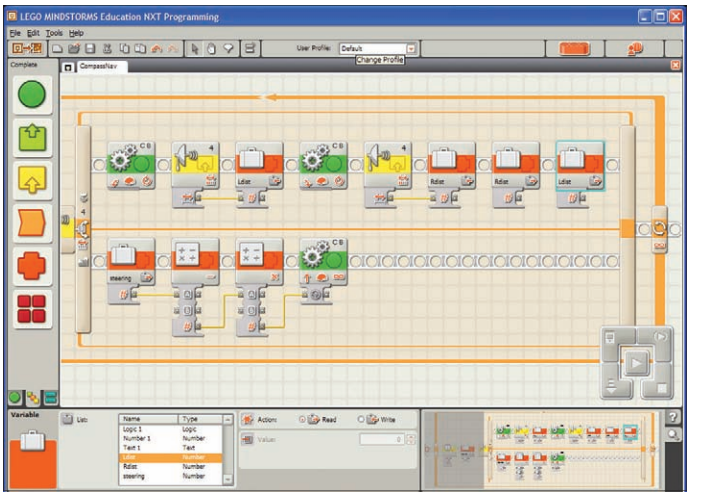


Figure 12. Add two variable blocks. Set the first to Rdist and the second to Ldist. Both should be set to read.

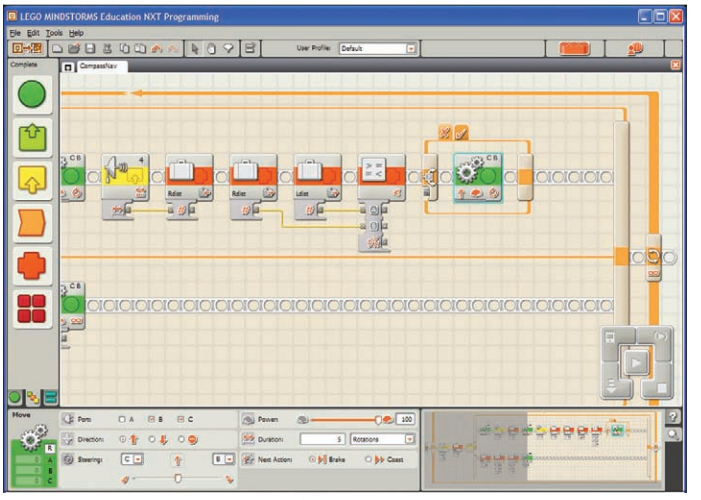


Figure 14. Add a switch with Control set to Value and Type set to Logic. Check Flat View, then select the X (False) tab on the switch. Add a move block going forward for five rotations. This means that if Eddie is already pointing to the clearest path, he'll immediately move forward for five rotations. Hopefully, that will be long enough to get around the obstacle.

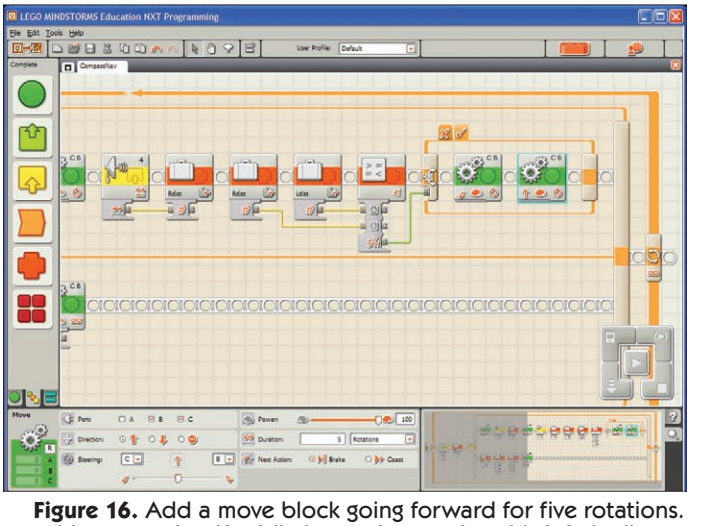


Figure 16. Add a move block going forward for five rotations. This means that if Eddie has a clear path to his left, he'll turn left, then go forward for five rotations.

Eddie gets too close. This will make him much more effective at avoiding obstacles.

- Try using two ultrasonic sensors: one to the left and one to the right. Have Eddie go forward until one of them gets under a threshold, then navigate around whatever Eddie sees.

Recap

In this edition, we used a compass sensor and an ultrasonic sensor to navigate obstacles while maintaining a course. Next month, we'll be using two awesome components available from **HiTechnic.com**: the NXT IRSeeker V2 and HiTechnic Infrared Electronic Ball. Hop on over to their website if you'd like to pick them up.

Until next month, this is Greg "LEGO" Intermaggio, signing off. **SV**

Then and NOW

WHAT DOES IT TAKE TO BUILD A ROBOT?

by Tom Carroll

That question might sound a lot like 'How to Build a Robot,' but I want to look a bit deeper into why people build robots. I've asked many robot experimenters just what it took for them to get started in robotics and I got the same number of different answers back. Yes, it is a very broad question, but I like to take time with different people who ask me the same thing to explain robotics to them and get them interested in the science or hobby. Mostly, I like to find out a bit about their background and possible reasons for wanting to build a robot.

Many might tell me "I know quite a bit about programming but little about mechanical things. Can I build a nice robot?" Yes! "I'm good with my hands and like to build all sorts of mechanical things but I don't know much about computers. Can I build a robot?" Yes! "My daughter is interested in science and has expressed interest in robots. Is there any way that I can help get her started in building robots?" Certainly! "I saw some combat robots on TV bashing each other's brains out and I thought that was really cool. Can I do that?" Of course!

Needless to say, if someone is asking a question like that, they need some encouragement, information, and possibly someone who is knowledgeable in robotics to mentor them at the start. I remember hearing people ask me why I was interested in robotics way back in the '60s when I was a kid and there were very few 'how to' books available on robot building. I didn't really need a good reason *why* I wanted to build a robot, I just did. Now that I'm quite a bit older, and speak and write about robots, I've had to think about what it really takes to build a robot. It basically boils down to the *desire* to build a robot, the required *materials*, and the *knowledge* of how to proceed.

The Early Days Of Robot Building

I'm going to refer to some of my early experiences and projects in this article as examples of early robot experimentation. Maybe my upbringing in a small North Carolina country town led me to devise ways to convert junk, mechanical 'who knows whats', and trashed appliances into robots, as there were no RadioShacks, Lowes, or other sources of good robot parts anywhere nearby. If someone had asked me why I wanted to build a robot back when I was in the eighth grade, I would have answered I have no real reason, it just sounds cool to be able to build one like I saw in a newspaper article. My older brother was going to NC State in the big city of Raleigh 60 miles away and could get me some old junked juke box and pinball machine parts, miscellaneous relays, and wire or other surplus things, but I was left on my own to figure out how to build a robot. With no Internet invented yet (and no computers) and a town library not much larger than the size of my garage, robot information was pretty much non-existent. I certainly had some obstacles ahead of me — the (seemingly) complexity of the various designs I'd considered and the high costs of materials, junk, or otherwise.

All it took was a few *Popular Mechanics* or *Popular Science* articles to let me realize if someone else could build a robot, so could I. However, in those days, what it took to build a robot was whatever other experimenters or I could scrounge up and an inventive imagination. Gone are the *true* days of experimentation before my time where young people could follow careful instructions from *Mechanics Illustrated* or similar magazines to construct a robot, radio, or some unique motorized project that was not a kit. *Popular Mechanics* featured a series of articles in their magazine called the *Boy Mechanic* at the turn of the 20th century. A series of reprints including the *Boy Scientist* shown in **Figure 1** instructed youngsters in how to build almost anything. With a post-Victorian thought process in those days, unfortunately girls were left to pursue other interests. In pre-WWII days, these magazines quite often featured construction projects that specified parts that could be found around most homes. Tin cans, old metal skate wheels, wheels from toys, dry-cell batteries, and small

DC motors from cars were the basis for some great projects for both kids and adults. Many projects instructed people on how to construct their own DC motors to power projects.

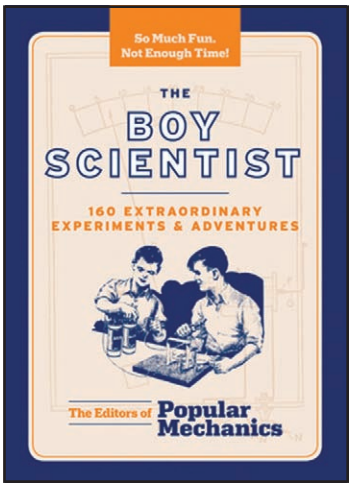
Most of the robots I remember reading about as a kid were either full-sized humanoid robots that could not walk or experimenter's smaller creations that rolled around on the floor like Grey Walters' Elsie. Major universities around the country had built what I felt in the '40s and '50s, were absolutely amazing robots — robotic creations that had crude intelligence and sensors, at least compared with today's technology. Rather than having today's programmed microcontroller delivering signals to parts of a robot, cams and relays drove motors, lights, and solenoids in early robots. Various branches of the military were experimenting with autonomous or remote-controlled planes, tanks, and boats, and had budgets far beyond what a typical home experimenter had. None of what anybody else was building really mattered though, as each builder had his (or her) own agenda. Everybody knew exactly what they wanted to build and why.

Shakey And Other Real Robots Inspired Early Builders

Even at an early age, one did not need to be the proverbial rocket scientist to realize that movie robots were either very crude mockups of someone's crazy idea of what a robot should look like or someone in a very uncomfortable robot suit. These props were made to last only a single session on a sound stage and then were sent to a prop storage house or junked. With Sci-Fi movies aside, it was the *real* robots in the news headlines that inspired the true robot experimenters. One of these real robots that inspired me the most was Stanford Research Institute's Shakey. This amazing robot was built in 1966 and was used for robotics research until 1972. Shown in **Figure 2**, Shakey was the first mobile robot to reason about its actions and was developed by SRI's Artificial Intelligence Center.

I've written about or mentioned Shakey in several articles over the past seven years, not because I was interested in building a robot just like him, but because the robot was a research platform that could evolve over the years. I certainly did not have access to DEC PDP-10 and PDP-15 computers that served as Shakey's intelligence, but the structural layout of the base, the processor box above that, the yoke holding the TV camera and range finder, and the overall appearance seemed to appeal to me as a great design. **Figure 3** is a basic sketch of Shakey with a proposed retractable arm. When I first saw him wobbling about in person during a tour, I could easily see where the gangly robot got its name. As a development platform, Shakey was used to develop robot perception, real world modeling, route planning, and basic movement control with sensor inputs. In 2004, Shakey was sent to the Robot Hall of Fame at Carnegie Mellon University and later found its home at the Computer History Museum in Mountain View, CA.

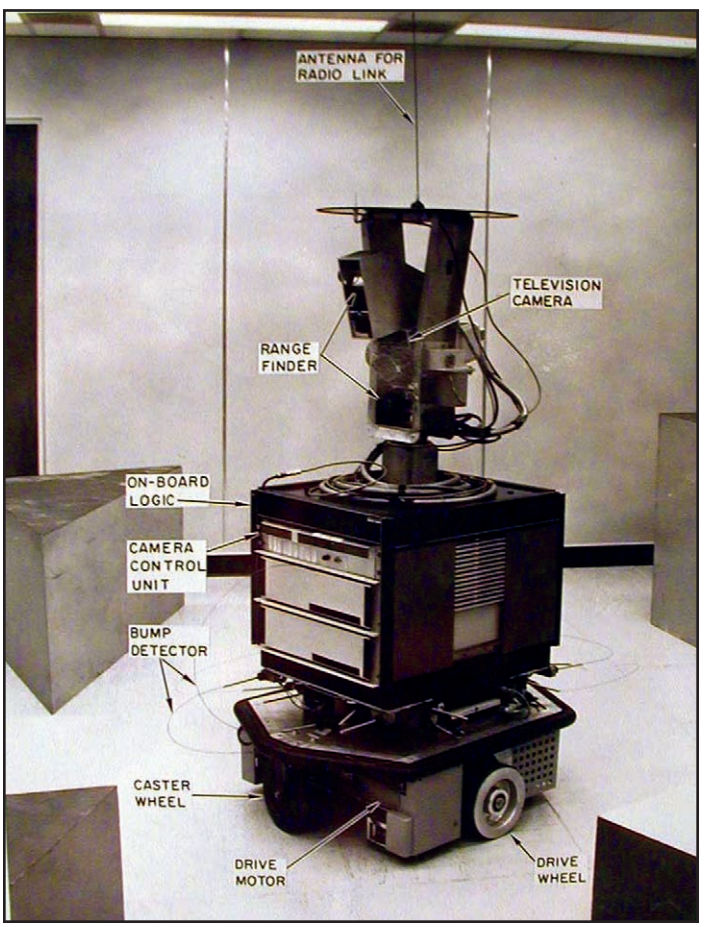
FIGURE 1. *Popular Mechanics'* Boy Scientist book.



The Stanford Artificial Intelligence Lab Cart Robot

A bit after Shakey, Stanford grad student Hans Moravec developed a unique robot in the late '70s for his Ph.D. thesis delivered in 1980. Hans and some other students built a card table sized robot shown in **Figure 4** at Stanford's Computer Science Department. I found this design to be inspiring in that it used an extremely simple design with four small bicycle tires, a couple of car batteries, and bicycle chains to steer the front wheels. Certainly Shakey — a decade earlier — had more sophisticated drive motors, on-board I/O processors, and was a bit more polished in looks, but functionality was Moravec's goal, not outward appearances. The cart that he used was actually borrowed from an earlier NASA-funded

FIGURE 2. SRI's Shakey the Robot in 1968.



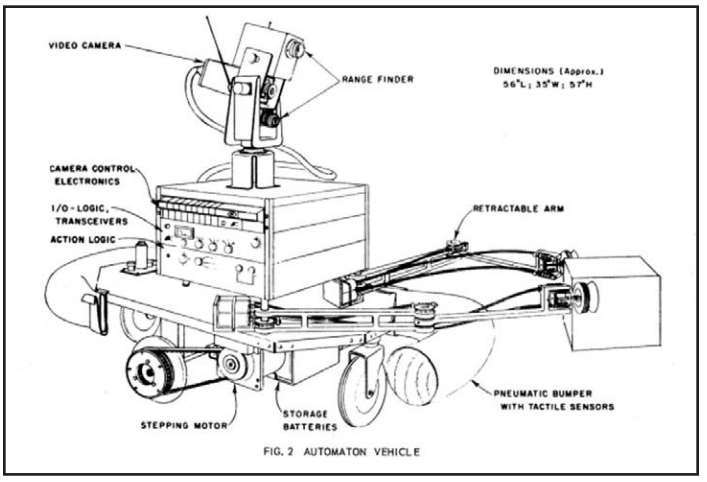


FIGURE 3. A sketch of a prototype Shakey with a retractable arm.

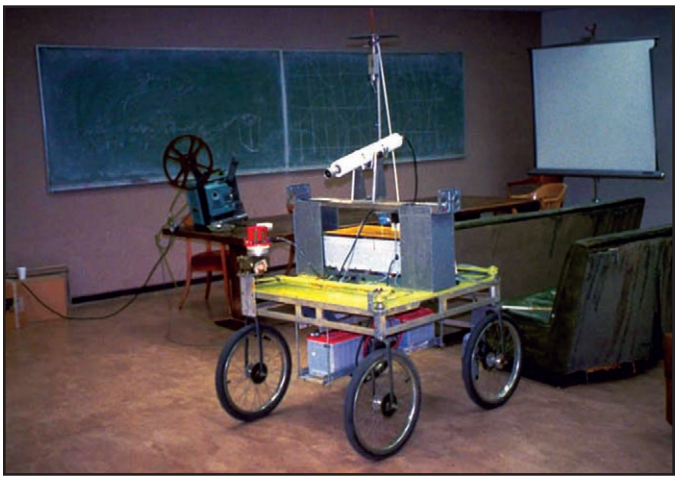


FIGURE 4. Stanford's Lab Cart robot in 1979.

project to develop an effective remote-controlled lunar robot.

His particular project was to develop a robotic platform that used video images derived from a single TV camera to plot its way about a cluttered environment. As with Shakey, the 'intelligence' was an off-board computer connected via an RF link. The computer was programmed to drive the cart around an unknown area, processing a series of stored TV images to gain knowledge of its world.

The only sensor was the single TV camera, and 3-D imagery was obtained by moving the camera back and forth via the cart's camera slider mechanism shown in **Figure 5**. There were no feelers or sonars, and only two wheels were steered Ackermann style. The cart developed 3-D locations of the several objects placed in its path, using its own motion and the side-to-side movement of the TV camera. As Moravec stated in his thesis: *"The system is moderately reliable, but very slow. The cart moves about one meter every 10 to 15 minutes, in lurches. After rolling a meter, it stops, takes some pictures, and thinks about them for a long time. Then, it plans a new path, and executes a little of it, and pauses again."*

One amusing scenario that I heard about was the confusion that the cart encountered when the shadows caused by the sun were distorted as the sun slowly moved across the sky in an outdoor setting (such as in **Figure 6**). I can just imagine the robot crying out, "What's going on, here! That shadow was square 15 minutes ago."

Fueling The Fires Of Inspiration

I used Shakey and the Stanford Lab Cart as examples of robots, not to inspire a budding robot builder to create a similar platform, but to describe two entirely different robots that experimenters used to examine some robot thought processes. These were university-level experiments. A parent can inspire budding robot interests in a youngster by introducing him or her to a school's LEGO club, buying them a subscription to *SERVO Magazine*, buying one of the learning kits from companies like Parallax with their Boe-Bot kit or the OLLO kit from Robotis.

Older kids can be encouraged to participate in high school level competitions like FIRST. An introduction to one of the many robot clubs across the country will allow kids with early interests in robotics to be paired with mentors with a lot of experience. All of these suggestions also apply to adults with an interest in this exciting science.

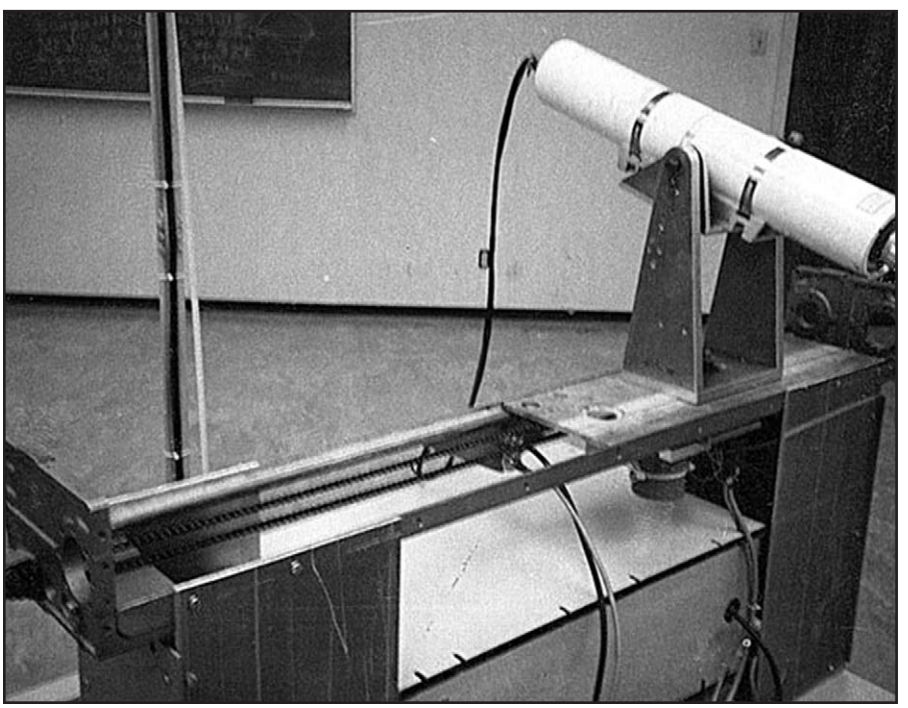


FIGURE 5. Cart camera slider mechanism in 1979.

Why Do I Desire To Build A Robot?

Going back about seven decades, the owners of a shop in New York City in 1933 were asking themselves that question when they decided to build a 'cowbot.' Cowbot? 1933? Messmore and Damon were not thinking of the term robot, but were specialists in building all sorts of mechanical animals from dinosaurs to fish and dogs. **Figure 7** depicts an exact mechanical reproduction of a Holstein milk cow. Headlines of the day stated: "Hidden Motors Give Exhibit for World's Fair the Movements of a Living Animal." Actually, only a single motor was used to allow the electric cow to move its jaws to chew its cud, breathe, move its head back and forth, moo, and actually give real milk. Covered with real cowhide over papier-mâché, a hatch covering the mechanisms was actually a black patch on the side of the cow that could be opened for servicing the many cams and gears. (Not bad for a $3,000 cow.)

FIGURE 6. Stanford Lab Cart outdoors in 1979.

The silent AC motor drove all the motions and a separate pump recirculated the milk as shown in the **figure**. A bellows produced the mooing sound and the sides of the cow moved in and out to simulate breathing. Reports revealed that it was one of the most watched exhibits at the fair with life-like movements every bit as believable as replicas that one might find in one of today's theme parks. The programming was strictly through the hand-carved rotating cams that drove idler wheels and levers to create all of the movements.

Looking again at the desire to build a robot, there might be several reasons why you want to undertake a robot project. Seventy years ago, you might have answered that question with: "I want to build a robot that looks like the cow I saw at the New York World's Fair." When Sci-Fi movies started to become popular 50 years ago, you saw real robots that caught your eye, or least they looked real. Of course, you pretty much knew that Robbie from *Forbidden Planet* or Tobor from *Tobor the Great* (shown in **Figure 8**) were fake humanoids, as was C-3PO decades later. It didn't take experimenters long to figure out that making a humanoid robot walk was virtually impossible in those days. Instead, you might have become tired of cleaning up after your cat or dog and decided to build a pet that only required charging. It didn't need to look anything like a cat or dog or even a cow — it just needed to please *you*.

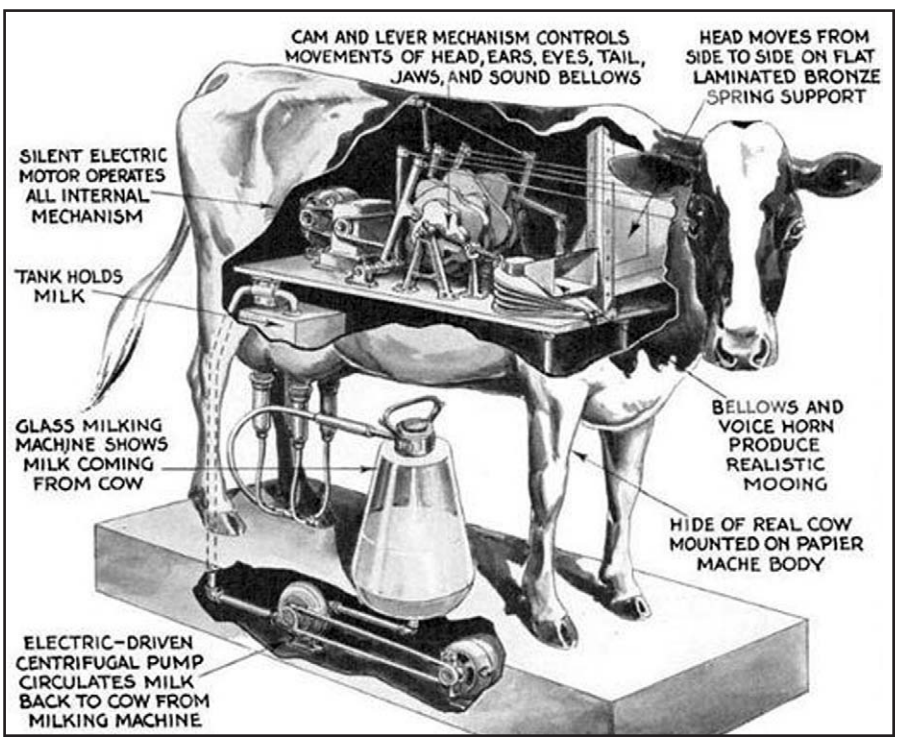


FIGURE 7. Cowbot drawing from the 1933 World's Fair.

Categories Of Home-Built Robots

There are many categories of robots that might be of interest to a home experimenter. Certainly, building a robot that you've seen in a movie might be a goal. More modern movies such as *Star Wars* and *Short Circuit* have been the

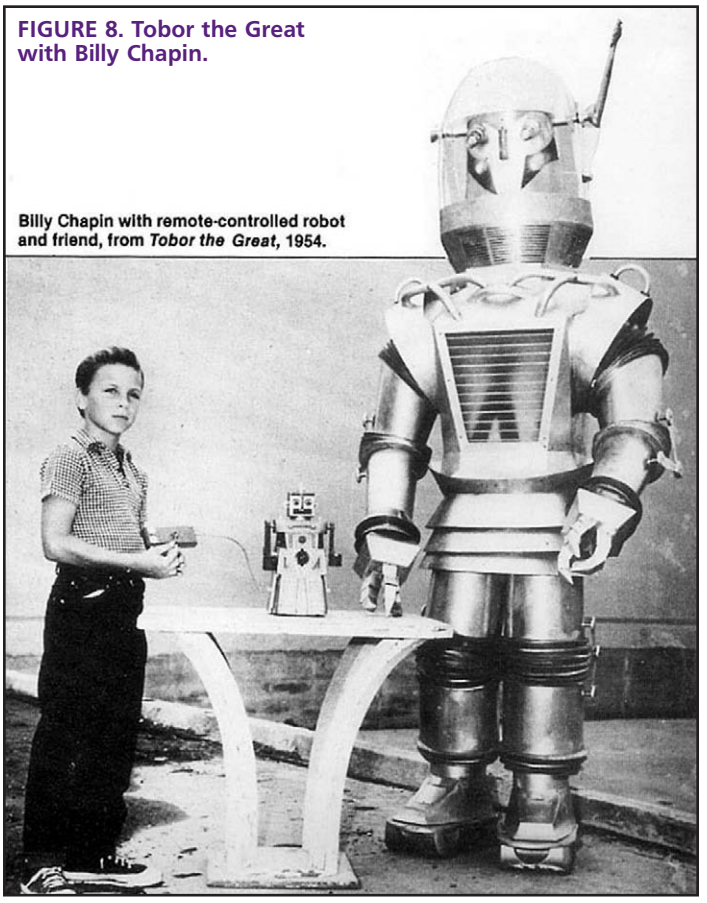


FIGURE 8. Tobor the Great with Billy Chapin.

FIGURE 9. LEGO Johnny Five.

sources of inspiration for thousands of robot builders. R2D2 is a favorite as is Johnny Five. A LEGO Johnny Five is shown in **Figure 9**. If a builder wants to adhere to the original's design and function, the can-shaped R2D2 is a bit easier to build, though the color schemes and decorations are hard to reproduce. Johnny Five is more technically challenging due to the tracks and various articulations of the body, arms, and head. Your levels of perfection are the only limitations in construction.

Testing a program, sensor, or mechanism might be your reason to build a robot. Test bed machines are frequently the goal of experimenters. You might have a loftier goal of producing an affordable law enforcement robot or a robot to serve the needs of the elderly. You might have received a nice R/C helicopter like I did this past Christmas and want to add some level of autonomy or a TV camera. Humanoids are now the rage, as are hexapods and quadrapods. Autonomous underwater vehicles and unmanned aerial vehicles are also becoming more popular. These are just a few of hundreds of types of robots, limited only by your imagination.

Do I Have The Right Materials To Build A Robot?

That is a loaded question, especially these days with so many manufacturers of robot parts and kits. In the early days of robot building, there wasn't all the various robot-specific materials available, but ingenuity abounded. Junkyards, abandoned cars, and old kitchen appliances still are full of great robot parts. Tin cans, forced-air heater ducts, toys, and even war surplus stuff were and are the basis of many robots. Plumbing fittings, cabinet, window, and door hardware were and are still quite useful. One can build a very functional robot with only basic materials.

Adding some model aircraft servos to provide extra motion, a BASIC Stamp, Arduino, or any of the multitude of inexpensive microcontrollers plus a few sensors will make your creation something that you can be proud of.

Final Thoughts

A lot of people who ask the title question may already know most of what it takes to build a robot. They may be in college and have a good background in programming and the use of microcontrollers, and all they need is a bit of encouragement to go out and tackle the mechanical part. Whether high school, college age, or adult, some people feel that robots are just computers with motors attached. In a way, it is that aspect that makes robotics so interesting, but it is also that aspect that makes robotics so complicated. Start with a small machine and learn with each step until you've reached a level that satisfies your desires. Whatever that desire might happen to be. **SV**

*For info on National Robotics Week April 9-17, go to **www.nationalroboticsweek.org***

Tom Carroll can be reached at TWCarroll@aol.com.

ROBO-LINKS

Advertiser Index

Now Serving Europe

with Optimized Logistics

- Currency in EURO
- 3 day shipping to 70% of the territory
- Service in French and English
- Competitive shipping rates
- Huge product selection

One Global Door for Manufacturers

VISIT: www.RobotShop.com

Shipping Worldwide

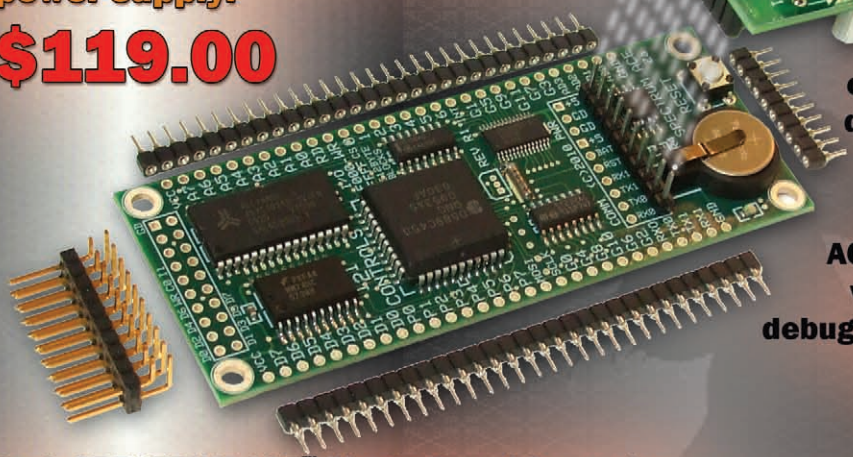
Robotics at your service! ™

I²I Controls

by engineers, for engineers

The SPECTRUM ACE™ 2a Starter Kit comes with the SPECTRUM ACE™ 2a CPU, all of the connectors shown, plus the SPECTRUM ACE™ Comm Board and a 5V DC 1 amp UL listed power supply.

$119.00



Our Advanced Control Environment (ACE) is composed of features that create an integrated solution for design, development, delivery and maintainance of even the most complex embedded systems.

ACE features ALEC™, our operating system with a built-in text editor plus integrated debugging and trace commands. This provides you with instant access to your code for debugging and optimization, whether on the bench or in the field.

Each SPECTRUM ACE™ Single Board Computer features I2I Controls' ALEC™ (Advanced Language for Embedded Control) operating system.

ALEC™ employs our Resident Run-Time Micro-Compilation process which produces highly compact code that typically executes between 30 and 40 thousand instructions per second.

The SPECTRUM LITE™ Evaluation Package comes with a SPECTRUM LITE™ CPU, the SPECTRUM LITE™ proto board featuring the 4 channel, 16-bit 1-wire DS2450, and a 5V DC 1 amp power supply.

$89.00

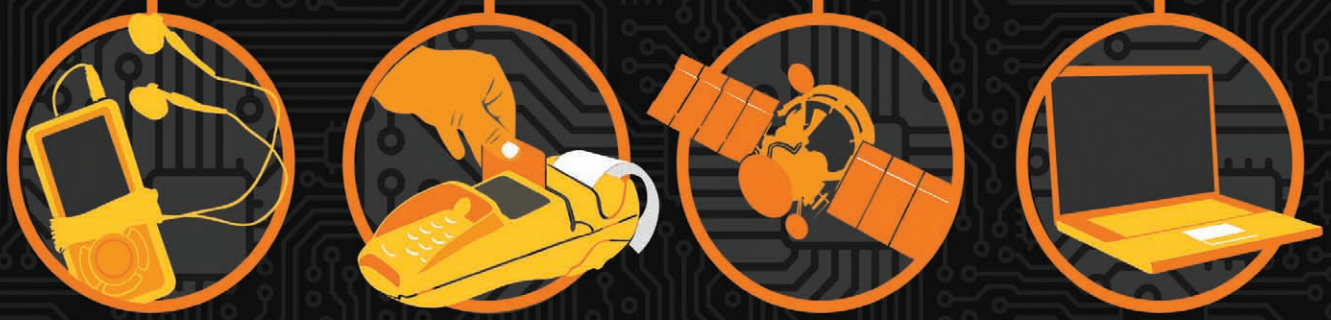
Edit your code in SRAM, then store your programs in the on-board FLASH or to an external USB drive. Since ALEC™ supports multiple boot modes, the user can opt to boot the system from SRAM, FLASH or USB. USB flash drives are supported with a suite of DOS-like disk instructions, providing storage for both bootable programs and/or run-time data. The USB Interface board, along with many other add-on devices, can be ordered from our website.

www.i2icontrols.com

NO MATTER WHAT THE IDEA

YOUR PCB PROTOTYPES SHOULD BE THE EASY PART

QUOTE & ORDER PCBS ONLINE AT **WWW.SUNSTONE.COM** OR CALL **1-800-228-8198**



SUNSTONE CIRCUITS

THE EASIEST PCB COMPANY TO DO BUSINESS WITH

ValueProto™ PCBexpress® Full Feature

Sunstone Circuits® pioneered the online ordering of printed circuit boards and is the leading PCB solutions provider with more than 35 years of experience in delivering quality prototypes and engineering software. With this knowledge and experience, Sunstone is dedicated to improving the PCB prototyping process from quote to delivery (Q2D®).

Did You Know? Sunstone Offers:

- Controlled impedance testing
- Fine lines and spacing [.003]
- RoHS compliant finishes
- Free 25-point design review
- Free shipping & no NRE's
- Flex / Rigid Flex Boards
- Online Quote & Order
- PCB123® design software
- RF / Exotic Materials
- Over 99% on-time or early delivery
- Best PCBs in the industry
- Live customer support 24/7/365